INPUT
1
CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER
DE AGOSTINI
A SOLE LIRE
2800

CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER

Direttori: Achille Boroli - Adolfo Boroli

Direzione editoriale: Mario Nilo; *settore fascicoli:* Jason Vella

Redazione dell'edizione italiana a cura della: Logical Studio Communication
Traduzione dall'inglese a cura di: Daniel Quinn

Coordinamento grafico: Otello Geddo
Coordinamento fotografico a cura del Centro Iconografico dell'Istituto Geografico De Agostini
Direzione: Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5
Redazione: Milano (20149), via Mosè Bianchi 6 - tel. (02) 4694451

Programma di abbonamento. Condizioni di abbonamento all'intera opera in 52 fascicoli, completa di copertine e di risguardi per la confezione dei 6 volumi dell'opera:
a) in un unico versamento anticipato di L. 180 000 in Italia, L. 225 000 all'estero;
b) in 4 versamenti trimestrali consecutivi e anticipati di L. 45 250 ciascuno.
La forma di abbonamento *b* è ammessa soltanto in Italia.
Agli abbonati all'intera opera sono riservate in dono "2 cassette di videogiochi" oppure, in alternativa, "5 cassette da registrare" (Aut. Min. conc.).
I versamenti possono essere effettuati a mezzo assegno bancario oppure sul c/c postale n. 111286 intestato all'Istituto Geografico De Agostini - Novara.
Amministrazione, abbonamenti e servizio arretrati: Istituto Geografico De Agostini - Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5.
Copertine e risguardi per i volumi dell'opera saranno messi in vendita a L. 6000 la copia (L. 7500 all'estero).
Le copie arretrate saranno disponibili per un anno dal completamento dell'opera e potranno essere prenotate nelle edicole o direttamente presso l'Editore. Per i fascicoli arretrati, trascorse 12 settimane dalla loro pubblicazione, è applicato un sovrapprezzo di L. 400 sul prezzo di copertina in vigore al momento dell'evasione dell'ordine.
Spedizione contro rimessa di pagamento anticipato; non vengono effettuate spedizioni contrassegno.
L'Editore si riserva la facoltà di modificare il prezzo nel corso della pubblicazione, se costretto da mutate condizioni di mercato.

Registrato presso il Tribunale di Novara n. 11 in data 19-5-1984.
Direttore responsabile: Emilio Bucciotti

Spedizione in abbonamento postale Gruppo II/70 (Autorizzazione della Direzione provinciale delle PP.TT. di Novara).
Distribuzione A. & G. Marco - Milano, via Fortezza 27 - tel. (02) 2526.
Pubblicazione a fascicoli settimanali. Esce il martedì.
Stampato in Italia - I.G.D.A. Officine Grafiche, Novara - 198410.

Referenze dei disegni e delle fotografie:
Copertina: Ian Taylor. Pagg. 8, 9, 12, 13, 15 Jeremy Gower. Pagg. 16-21, disegni: Peter Bentley; foto: John Darling. Pagg. 22-23 (alto) Kevin O'Keefe, (basso) Chris Lyon. Pagg. 24-25 Kevin O'Keefe. Pag. 27, disegno: Chris Lyon; foto: NASA. Pagg. 28-29, disegno: Chris Lyon; foto: Tony Stone Associates. Pagg. 30-31, disegno: Chris Lyon; foto: Jerry Young. Pag. 32, disegno: Chris Lyon; foto: Rex Features Ltd. Effetti al computer della J. D. Audio Visual.

Con questo fascicolo è data in omaggio la cassetta "PROGRAMMA CHAMP" (Aut. Min. conc.).

Pubblicazione a fascicoli settimanali
edita dall'Istituto Geografico De Agostini

volume I - fascicolo 1

PROGRAMMAZIONE BASIC 1

PENSA UN NUMERO **2**

La funzione RND. IF ... THEN. Variabili. INPUT

CODICE MACCHINA 1

COME RENDERE PIÙ VELOCI I GIOCHI **8**

Qualche effetto grafico per avvicinarsi al codice macchina

PROGRAMMAZIONE BASIC 2

L'ARTE DI USARE I CICLI *FOR ... NEXT* **16**

Il computer funziona da contatore

PERIFERICHE

COME USARE LE *SAVE* E LE *LOAD* **22**

Utili consigli sull'uso dei registratori a cassette
per memorizzare programmi e dati

GIOCHI AL COMPUTER 1

QUALCHE MOSSA D'ANIMAZIONE **26**

Curiose immagini con pochi elementi grafici

INPUT È STUDIATA APPOSITAMENTE PER:

Lo SPECTRUM della Sinclair (versioni 16K e 48K), il COMMODORE 64, l'ELECTRON ed il BBC della ACORN, il DRAGON 32.

Comunque, molti dei programmi e dei testi sono adatti anche per: lo ZX81 della SINCLAIR, il COMMODORE VIC 20 ed il TANDY COLOUR COMPUTER con 32K ed il BASIC esteso.

I seguenti simboli identificano i programmi o le spiegazioni adatte a ciascun computer:

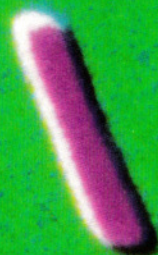
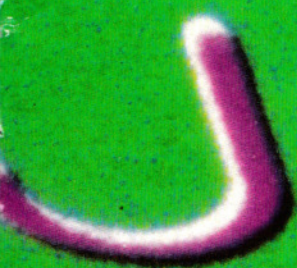
 SPECTRUM

 COMMODORE 64

 ELECTRON e BBC

 DRAGON 32

 ZX81 VIC 20

 TANDY TRS80 COLOUR COMPUTER

f8
f9
I
INPUT
U
K
RAD
CH

PENSA UN NUMERO...

La maggior parte dei giochi usa la generazione di numeri casuali per creare sequenze di eventi apparentemente irregolari.

Imparare a programmare è un po' come imparare il gioco del calcio.

Teoricamente, si potrebbe studiare ogni singola mossa sulla carta e scendere in campo soltanto quando si è in grado di padroneggiarle tutte, ma ciò, oltre ad essere scarsamente divertente, sarebbe anche molto faticoso e lento.

Lo stesso si può dire della programmazione: si possono leggere e rileggere un'infinità di testi, ma il metodo migliore è proprio quello di 'entrare in campo'. Da dove inziare?

INDOVINARE UN NUMERO

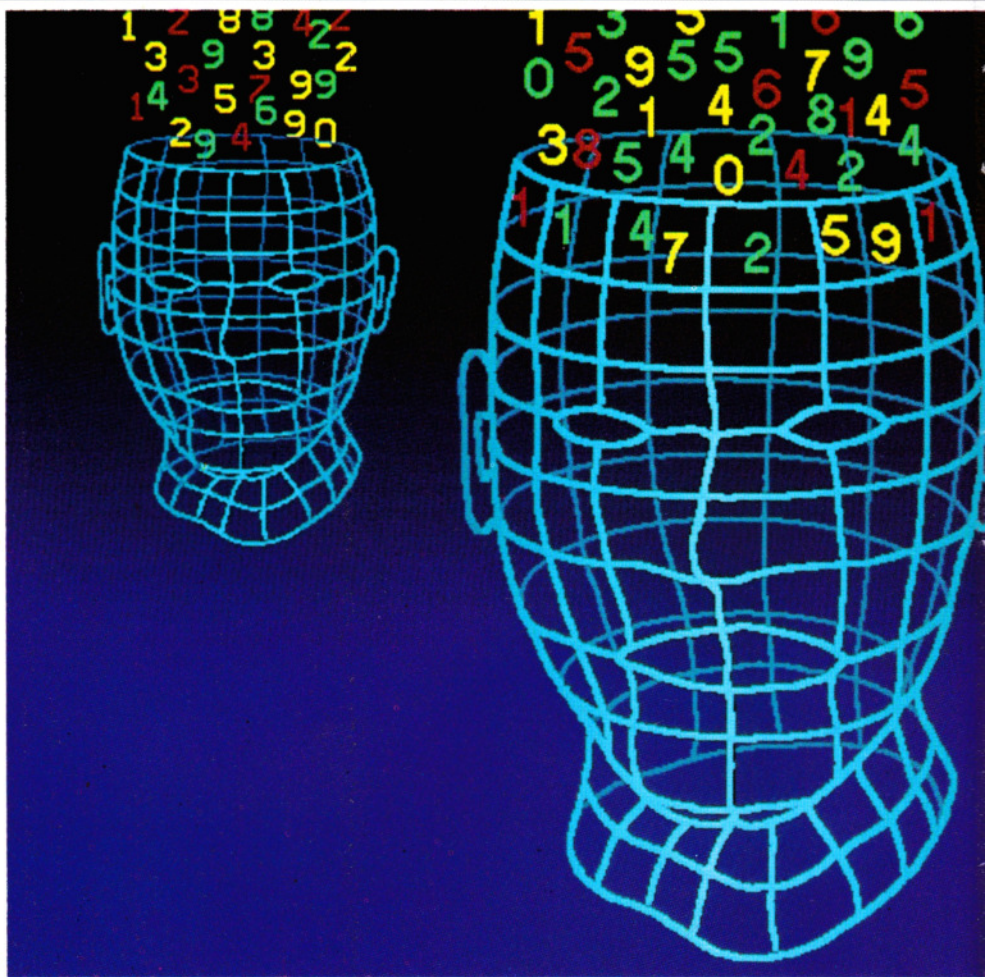
Il più semplice gioco per computer è quello nel quale la macchina 'pensa' un numero a caso e il giocatore deve indovinarlo.

Su alcuni computer si usa l'istruzione RANDOMIZE A cosa serve?

Sullo Spectrum, usando RANDOMIZE 1 (o qualsiasi altro numero), il computer genera sempre la stessa sequenza di numeri casuali.

Questa circostanza si rivela utile durante la ricerca di eventuali errori nel programma.

Sugli altri computer, è possibile ottenere il medesimo effetto per mezzo dell'istruzione RND (−1). Anche in questo caso, il valore del numero, purché negativo, non è rilevante.

Usando, sullo Spectrum, RANDOMIZE Ø o omettendo del tutto il numero, si ottiene l'effetto opposto, ossia che le sequenze dei numeri siano quanto più casuali possibile.

LA FUNZIONE RND

Tutti gli home computer sono dotati di un *generatore casuale di numeri*, utile in questo genere di giochi. In BASIC vi si accede mediante la funzione RND.

I numeri prodotti, tuttavia, non sono nella forma per noi più conveniente, essendo dei valori frazionari compresi tra Ø e Ø,99999999. Verifichiamolo con un semplice programma, preceduto da un comando NEW, onde ripulire la memoria da eventuali programmi preesistenti:

```
10 LET X=RND
20 PRINT X
30 GOTO 10
```

```
10 LET X=RND (Ø)
20 PRINT X
30 GOTO 10
```

```
10 LET X=RND (1)
20 PRINT X
30 GOTO 10
```

(Ricordarsi di premere il tasto RETURN o ENTER al termine di ciascuna linea di programma).

Adesso, impartendo un comando RUN, tutto ciò che vedremo sarà un'interminabile serie di numeri frazionari (si veda, a tal proposito, la tabella a pagina 7), poco utili per il nostro gioco.

Per ovviare a questo incoveniente, viene impiegata la funzione INT (abbreviazione di *intero*) e il valore viene opportunamente moltiplicato:

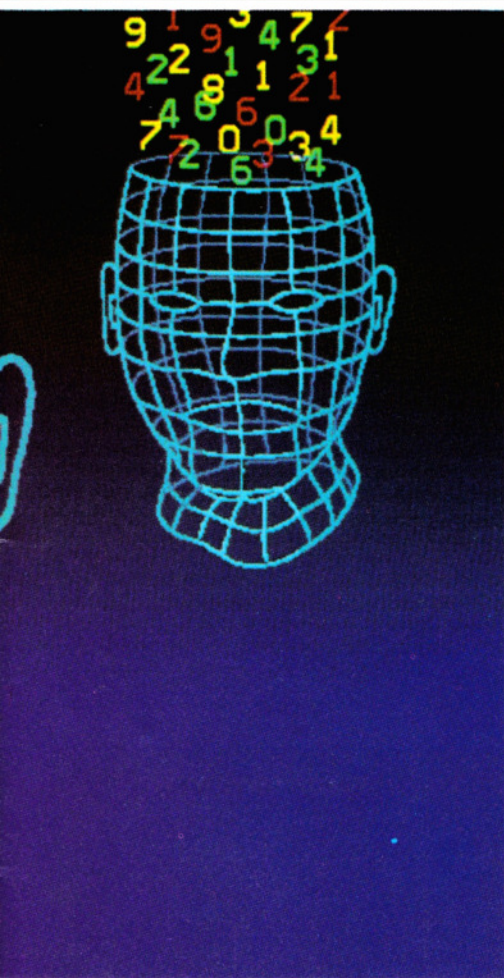
- LA FUNZIONE **RND**
- COME GENERARE NUMERI CASUALI
- USO DELLE VARIABILI
- L'ISTRUZIONE **INPUT**
- L'USO DI **IF ... THEN** NEI CONFRONTI
- DUE PROGRAMMI BASATI SUI NUMERI
- LA GAMMA DEI NUMERI CASUALI

```
10 LET X=INT (RND*6)
```

```
10 LET X=INT (RND(1)*6)
```

In questo caso, i numeri generali sono compresi tra Ø e 5.

Su alcuni computer non occorre usare la INT, ad esempio:

```
10 LET X=RND (6)—1
20 PRINT X
30 GOTO 10
```

(Ricordarsi di premere il tasto RETURN o ENTER, al termine di ciascuna linea).

Qualsiasi computer si possieda, l'arco di valori non è limitato a quelli tra Ø e 5, ma il gioco sarebbe arduo se scegliessimo una gamma tra 1Ø e 1Ø ØØØ!

USO DELLE VARIABILI

Nei brevi programmi appena presentati, non solo abbiamo generato un numero, ma gli abbiamo anche assegnato un nome: X.

Da questo momento in poi, se adoperiamo di nuovo X, il computer sa di dover fornire il numero 'scelto a caso'.

Questa X, che serve al computer per identificare un particolare numero e usarlo in operazioni aritmetiche oppure in un confronto, si chiama *variabile.*

L'ISTRUZIONE INPUT

Avendo generato il numero casuale, il passo successivo è quello di far si che il computer accetti un nostro tentativo d'indovinarlo. A questo scopo ci serviamo dell'istruzione INPUT.

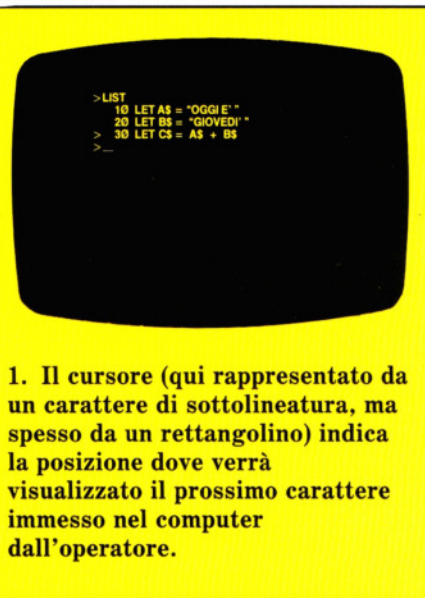
La sola parola INPUT non basta: occorre specificare anche un nome di variabile, nella quale il computer depositerà il valore immesso. Liberi di scegliere una lettera, o anche una combinazione di lettere qualsiasi (purché non X, già impegnata), usiamo una G.

L'istruzione, per intero (da non immettere ancora), è quindi:

```
INPUT G
```

Adesso, però, è necessario che il computer confronti il valore da lui generato con quello da noi immesso. Per far ciò impieghiamo una combinazione particolare del BASIC:

```
IF X=G THEN PRINT "RISPOSTA ESATTA"
```

La frase IF ... THEN (*se ... allora*) è molto utile e frequentemente impiegata nei programmi.

1. Il cursore (qui rappresentato da un carattere di sottolineatura, ma spesso da un rettangolino) indica la posizione dove verrà visualizzato il prossimo carattere immesso nel computer dall'operatore.

In alcuni computer (Acorn e Dragon), ne esiste una versione più sofisticata, ossia IF ... THEN ... ELSE (*se ... allora ... altrimenti*), la quale fornisce un'alternativa qualora la condizione *non* sia soddisfatta.

Negli altri computer, privi di ELSE (Spectrum e Commodore), se la condizione non è soddisfatta, il programma passa semplicemente alla linea successiva.

IMPOSTAZIONE DEL PROGRAMMA

Adesso, immettiamo il programma per intero: (si noti che < > significa rispettivamente 'minore di e maggiore di', in altre parole *'diverso da'*).

```
20 LET X=INT (RND*6)
30 PRINT "IL COMPUTER HA SCELTO UN
   NUMERO TRA LO Ø ED IL 5. RIESCI AD
   INDOVINARLO?"
40 INPUT G
60 IF G=X THE PRINT "RISPOSTA ESATTA"
80 IF G <> X THEN PRINT
   "ERRATO - RIPROVARE"
```

```
20 LET X=RND(6)—1
```

```
30 PRINT "IL COMPUTER HA SCELTO UN
   NUMERO TRA LO 0 ED IL 5. RIESCI AD
   INDOVINARLO?"
40 INPUT G
60 IF G=X THEN PRINT "RISPOSTA
   ESATTA" ELSEPRINT "ERRATO—
   RIPROVARE"
```

```
20 LET X=RND(6)—1
30 PRINT "IL COMPUTER HA SCELTO UN
   NUMERO TRA LO 0 ED IL 5. RIESCI AD
   INDOVINARLO?"
40 INPUT G
60 IF G=X THEN PRINT "RISPOSTA
   ESATTA" ELSE PRINT "ERRATO—
   RIPROVARE"
```

Una via d'uscita

Talvolta, i principianti trovano difficoltà nell'interrompere un programma in corso d'esecuzione, magari per tornare a lavorare sul listato. Ecco alcuni consigli su come procedere:

In primo luogo, si premano, contemporaneamente, CAPS SHIFT e SPACE. Se questo non dovesse funzionare, è molto probabile che il computer stia eseguendo un istruzione INPUT. In questo caso, usare i tasti per il movimento del cursore e il tasto DELETE per rimuovere eventuali apici o doppi apici. Successivamente, premere STOP (SHIFT+A), seguito da ENTER. Sullo schermo dovrebbe apparire la scritta 'stop in INPUT'. Adesso si prema ENTER per esaminare il listato.

Si provi premendo il tasto RETURN, seguito dal comando LIST. Se questo non funziona, allora tenendo premuto RUN/STOP, usare il tasto RESTORE. Poi usare il comando LIST.

Premere ESCAPE, poi digitare il comando LIST. Se non funziona: premere BREAK, digitare OLD, seguito da LIST.

Premere BREAK, poi digitare il comando LIST. Se non funziona: premere il tasto RESET, quindi usare il comando LIST.

```
20 LET X=INT(RND(1)*6)
30 PRINT "IL COMPUTER HA SCELTO UN
   NUMERO TRA LO 0 ED IL 5. RIESCI AD
   INDOVINARLO?"
40 INPUT G
60 IF G=X THE PRINT "RISPOSTA ESATTA"
80 IF G<>THE PRINT «ERRATO
   —RIPROVARE"
```

Impartendo un RUN, il programma funziona, ma il gioco si esaurisce in una sola 'mandata' e sullo schermo rimangono scritte superflue.

A questo secondo inconveniente si ovvia facilmente, basta aggiungere:

```
10 CLS
50 CLS
```

```
10 PRINT "♥"
50 PRINT "♥"
```

Questo accorgimento serve per 'ripulire' lo schermo prima di eseguire il resto del programma.

Per ottenere più di un 'lancio', invece, la cosa si presenta leggermente più complicata. Per il momento possiamo usare, semplicemente:

```
90 GOTO 10
```

che fa ripartire automaticamente il programma.

LE VARIABILI ALFANUMERICHE

Un metodo migliore, anche se apparentemente più complicato, consiste nel chiedere al giocatore se intende continuare nel gioco. Iniziamo coll'immettere il programma per intero (sullo ZX81, tralasciare OR A$="s" nella linea 110:

```
10 CLS
20 LET X=INT (RND*6)
30 PRINT "IL COMPUTER HA SCELTO UN
   NUMERO TRA LO 0 ED IL 5. RIESCI AD
   INDOVINARLO?"
40 INPUT G
50 CLS
60 IF G=X THEN PRINT "RISPOSTA
   ESATTA"
70 IF G=X THEN GOTO 90
80 IF G<>X THEN PRINT "ERRATO
   —RIPROVARE"
90 PRINT "PER UN ALTRO TENTATIVO
   DIGITARE UNA S, SEGUITA DA ENTER"
100 INPUT A$
110 IF A$="S" OR A$="s" THEN GOTO 10
120 GOTO 100
```

```
10 CLS
20 LET X=RND(6)—1
30 PRINT "IL COMPUTER HA SCELTO UN
   NUMERO TRA LO 0 ED IL 5. RIESCI AD
   INDOVINARLO?"
40 INPUT G
50 CLS
60 IF G=X THEN PRINT "RISPOSTA
   ESATTA" ELSE PRINT "ERRATO
   —RIPROVARE"
90 PRINT "PER UN ALTRO TENTATIVO
   DIGITARE UNA S, SEGUITA DA ENTER"
100 INPUT A$
110 IF A$="S" THEN GOTO 10
120 GOTO 100
```

```
10 CLS
20 LET X=RND(6)—1
30 PRINT "IL COMPUTER HA SCELTO UN
   NUMERO TRA LO 0 ED IL 5. RIESCI AD
   INDOVINARLO?"
40 INPUT G
50 CLS
60 IF G=X THEN PRINT "RISPOSTA
   ESATTA" ELSE PRINT "ERRATO
   —RIPROVARE"
90 PRINT "PER UN ALTRO TENTATIVO
   DIGITARE UNA S, SEGUITA DA ENTER"
100 INPUT A$
```

2. L'importanza della punteggiatura in un programma. Le linee nella parte alta dello schermo sono le istruzioni dell'esecuzione: la virgola (,) serve per 'tabulare', il punto e virgola (;) per una visualizzazione compattata, senza separazioni, mentre un apice ('), nel Commodore, nello Spectrum e negli Acorn, serve per passare alla successiva riga dello schermo.

```
110 IF A$="S" THEN GOTO 10
120 GOTO 100
```

```
10 PRINT "♥"
20 LET X=INT (RND(1)*6)
30 PRINT "IL COMPUTER HA SCELTO UN
   NUMERO TRA LO 0 ED IL 5. RIESCI AD
   INDOVINARLO?"
40 INPUT G
50 PRINT "♥"
60 IF G=X THEN PRINT "RISPOSTA
   ESATTA": GOTO 90
80 PRINT "ERRATO—RIPROVARE"
90 PRINT "PER UN ALTRO TENTATIVO
   DIGITARE UNA S, SEGUITA DA ENTER"
100 INPUT A$
110 IF A$="S" THEN GOTO 10
120 GOTO 100
```

Come si può notare, prima si chiede al giocatore se desidera giocare ancora (linea 90) poi, per immettere la risposta, si adopera una INPUT nella linea 100.

Questa volta, però, c'è una differenza sostanziale: alla linea 40 viene richiesta l'immissione di un valore numerico, mentre qui, invece, occorre digitare una 'S' o una 'N', ossia un carattere alfabetico. Ciò significa che dobbiamo adoperare un diverso tipo di variabile, adatto a contenere lettere (o parole intere) invece di numeri: scriviamo, pertanto, INPUT A$.

Il simbolo del dollaro comunica al computer che la variabile chiamata A deve contenere lettere e A$ è chiamata una *variabile 'stringa'*.

La differenza tra i due tipi di variabili diverrà più chiara in seguito. Per adesso basta ricordare che: per immettere *valori numerici* si usa INPUT A, INPUT B, INPUT X, e così via, mentre per immettere *lettere* o *parole*, occorre usare INPUT A$, INPUT B$, INPUT X$, e così via.

La linea 120 serve per ripetere la domanda (e l'immissione della risposta, qualora questa sia diversa da 'S').

Infatti, l'esecuzione viene dirottata alla linea 100 finché il carattere immesso dalla tastiera non corrisponde a una S (maiuscola o minuscola a seconda del programma). In tal caso, il programma riparte dall'inizio con un nuovo indovinello.

IMPARIAMO LE MOLTIPLICAZIONI

La funzione RND si rivela utile in molteplici occasioni. Supponiamo di voler insegnare a un bambino la tabellina del 9. Potremmo scrivere un programma così concepito:

```
10 PRINT «QUANTO FA 1 PER 9?"
20 INPUT A
30 IF A=9 THEN PRINT "CORRETTO"
40 PRINT "QUANTO FA 2 PER 9?"
50 INPUT B
60 IF B=18 THEN PRINT "CORRETTO"
```

... e così via, ma esso sarebbe decisamente chilometrico e non prevederebbe nemmeno il caso di una risposta errata!

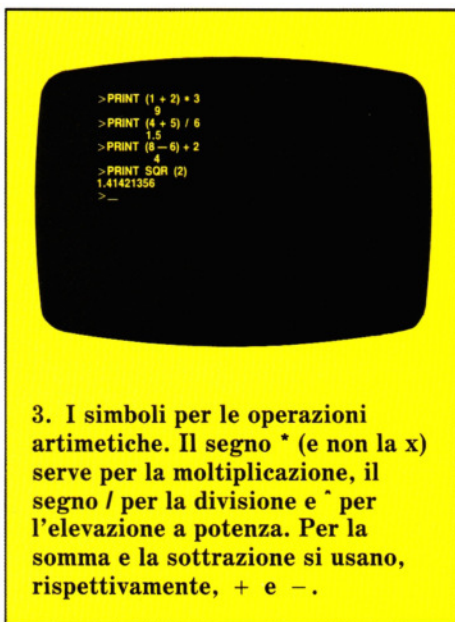
Un metodo migliore, applicabile molto più in generale, è basato sull'impiego della funzione RND, con la quale, oltretutto, è possibile evitare una precisa sequenza nelle domande, ottenendo uno strumento didattico di maggior efficacia.

Nella stesura di un programma è sempre bene partire da un nucleo centrale, lasciando gli abbellimenti a un secondo tempo. Proviamo il seguente programma (digitando un NEW per cancellare eventuali programmi precedenti):

```
10 LET N=INT (RND*12+1)
20 PRINT "QUANTO FA□"; N; "□PER 9?"
30 INPUT A
40 IF A=N*9 THEN PRINT "CORRETTO"
```

```
10 LET N=RND(12)
20 PRINT "QUANTO FA □"; N; "□ PER 9?"
30 INPUT A
40 IF A=N*9 THEN PRINT "CORRETTO"
```

```
10 N=INT (RND(0)*12)+1
20 PRINT "QUANTO FA□"; N; "□PER 9?"
30 INPUT A
40 IF A=N*9 THEN PRINT "CORRETTO"
```

La RND viene usata in modo molto simile a quello impiegato per il precedente gioco. Nella linea 10 si assegna a una variabile di nome N (il nome, si ricordi, è scelto a nostro piacimento) un valore generato a caso.

3. I simboli per le operazioni artimetiche. Il segno * (e non la x) serve per la moltiplicazione, il segno / per la divisione e ^ per l'elevazione a potenza. Per la somma e la sottrazione si usano, rispettivamente, + e −.

Il numero generato, in questo caso, è compreso tra 1 e 12. (Sullo Spectrum e sul Commodore occorre sommare 1, altrimenti otterremmo valori tra 0 e 11).

Nella linea 20 si chiede al giocatore di moltiplicare per nove il numero generato dal computer. La linea 40 segnala al computer che il numero casuale va moltiplicato per 9 e che il risultato va confrontato con la risposta del giocatore. Se questa è giusta, viene visualizzato (grazie alla PRINT) il messaggio 'ESATTO'. Si provi a 'lanciare' il programma (impartendo il comando RUN): esso funzionerà solamente una volta. Per far in modo che esso continui, si può aggiungere:

```
50 GOTO 10
```

ma perché non far le cose per bene, ossia:

```
10 PRINT "CIAO. COME TI CHIAMI?"
20 INPUT A$
30 CLS
40 PRINT "CIAO,□";A$, "HO
   ALCUNE","DOMANDE PER TE"
50 PAUSE 200
60 CLS
70 LET N=INT (RND*12)+1
80 PRINT "QUANTO FA□";N; "□PER 9?"
90 INPUT A
100 IF A=N*9 THEN GOTO 150
110 CLS
120 PRINT A;"□?"
130 PRINT "SPIACENTE, RIPROVA"
140 GOTO 80
150 PRINT "CORRETTO,□"; A$, "ECCO LA
   PROSSIMA"
160 PAUSE 150
170 GOTO 60
```

```
1Ø PRINT "CIAO. COME TI CHIAMI?"
2Ø INPUT A$
3Ø CLS
4Ø PRINT "CIAO,□";A$: PRINT "HO ALCUNE
   DOMANDE PER TE"
5Ø FOR X=1 TO 6ØØØ:NEXT X
6Ø CLS
7Ø LET N=RND(12)
8Ø PRINT "QUANTO FA□";N;"□PER 9?"
9Ø INPUT A
1ØØ IF A=N*9 THEN GOTO 15Ø
11Ø CLS
12Ø PRINT A;"?"
13Ø PRINT "SPIACENTE, RIPROVA"
14Ø GOTO 8Ø
15Ø PRINT"CORRETTO,□";A$:
   PRINT "ECCO LA PROSSIMA"
16Ø FOR X=1 TO 4ØØØ:NEXT X
17Ø GOTO 6Ø
```

```
1Ø PRINT "CIAO. COME TI CHIAMI?"
2Ø INPUT A$
3Ø PRINT "♥"
4Ø PRINT "CIAO,□";A$: PRINT "HO ALCUNE
   DOMANDE PER TE"
5Ø FOR X=1 TO 2ØØØ:NEXT X
6Ø PRINT "♥"
7Ø N=INT(RND(1)*12)+1
8Ø PRINT "QUANTO FA□";N;□PER 9?"
9Ø A=Ø:INPUT A
1ØØ IF A=N*9 THEN GOTO 15Ø
11Ø PRINT "♥"
12Ø PRINT A;"□?"
13Ø PRINT "SPIACENTE, RIPROVA"
14Ø GOTO 8Ø
15Ø PRINT "CORRETTO,□";A$:PRINT "ECCO
   LA PROSSIMA"
16Ø FOR X=1 TO 2ØØØ:NEXT X
17Ø GOTO 6Ø
```

La funzione della maggior parte delle linee aggiunte appare evidente se si esegue il programma.

Si noti, comunque, che:
le linee 3Ø, 6Ø e 11Ø servono per ripulire lo schermo da tutte le scritte superflue e da eventuali risposte errate (per verificarlo, si provi a lanciare il programma dopo

4. I numeri posti all'inizio di ciascuna linea di programma sono molto importanti. Senza di essi, il computer, in BASIC, non saprebbe in quale ordine eseguire le istruzioni. È utile numerare le linee di 1Ø in 1Ø, permettendo, così, eventuali inserimenti successivi.

aver cancellato queste tre linee);
le linee 5Ø e 16Ø servono soltanto per assicurare delle pause nell'esecuzione e ciò si ottiene semplicemente 'facendo contare' il computer da 1 fino a un certo numero sufficientemente grande, prima di proseguire col programma (il metodo è spiegato nella seconda lezione).

Nel programma per lo Spectrum, le virgole nelle linee 4Ø e 15Ø sono state inserite unicamente per evitare che le parole vengano troncate alla fine di una riga di schermo. Nelle altre versioni, questo si ottiene aggiungendo un ':' e una ulteriore PRINT.

A questo punto, il programma ha un solo difetto: non si ferma più!

Per ovviare a ciò:

Premere [STOP], poi [ENTER].

Premere [BREAK].

Premere [ESCAPE].

Premere [RUN/STOP].

Modificare il programma per la tabellina del 5, del 7, o di qualsiasi altro numero, è di una facilità estrema.

Inoltre, rispetto alla prima versione, c'è il vantaggio che non occorre conoscere a priori le risposte esatte per scrivere il programma!

Come si specifica un arco di valori nella generazione dei numeri casuali?

RND sullo Spectrum, RND(1) sugli Acorn e RND (Ø) sul Dragon e sul Commodore, generano un numero casuale compreso tra Ø e Ø,999999. Se si desidera un arco di valori diverso, occorre moltiplicare il valore ottenuto con la funzione RND. Per esempio, moltiplicando per 4Ø, si ottiene un numero compreso tra Ø e 39,999999.

Volendo ottenere soltanto numeri *interi* (senza valori dopo la virgola), si usa la funzione INT e l'espressione risulta: INT (appropriata funzione RND*4Ø), che genera valori interi tra Ø e 39. La funzione INT elimina, infatti, la parte decimale del numero.

Se, infine, volessimo un numero compreso tra 1 e 4Ø, basta sommare 1 al valore appena ottenuto. Il Dragon e gli Acorn sono dotati di una scorciatoia: la funzione RND (4Ø), nel nostro caso, fornisce già un valore intero. Si osservi la tabella sottostante per i vari esempi.

La generazione di numeri casuali

Genera un numero tra Ø e Ø,999999	RND(1)	RND(Ø)	RND(Ø)	RND
Genera un numero tra Ø e n*Ø,999999	RND(1)*n	RND(Ø)*n	RND(Ø)*n	RND*n
Genera un numero tra −1Ø e +1Ø	RND(21)−11	INT(RND(Ø)*21)−1Ø	RND(21)−11	INT(RND*21)−1Ø
Genera un numero tra Ø e 39	INT(RND(1)*4Ø)	INT(RND(Ø)*4Ø)	INT(RND (Ø)*4Ø)	INT(RND*4Ø)
Genera un numero tra 1 e 4Ø	RND(4Ø)	INT(RND(Ø)*4Ø)+1	RND(4Ø)	INT(RND*4Ø)+1

COME RENDERE PIÙ VELOCI I GIOCHI

Il codice macchina non è dominio esclusivo degli esperti. Si può cominciare a impararlo, usando brevi sottoprogrammi che rendono più rapida l'esecuzione dei giochi in BASIC

La programmazione in linguaggio macchina, almeno a prima vista, appare molto difficile. Alla maggior parte dei possessori di home computer, il codice macchina appare come una sfilza di numeri incomprensibili.

Esiste, tuttavia, un sistema assai semplice per impararlo: iniziando ad aggiungere, ai normali programmi in BASIC, brevi sottoprogrammi in codice macchina. Tra l'altro, ciò renderà i giochi più rapidi e divertenti, che non usando il solo BASIC.

Il metodo grafico impiegato nei programmi che seguono (appositamente creati per INPUT, è del tipo 'combinato' BASIC-codice macchina: i risultati sono paragonabili a quelli dei giochi reperibili in commercio.

Più avanti nel corso, verrà spiegato come ottenere altri interessanti effetti grafici e costruire immagini del tutto nuove.

Per costruire le immagini grafiche riportate nelle figure 2 e 3, sono necessarie tre operazioni.

La prima è quella di definire, nella memoria del computer, una cornice o 'griglia', onde stabilire le dimensioni dell'immagine grafica. Inizialmente, la definizione avviene mediante una serie di *simboli grafici definiti dall'utente* (si veda qui sotto).

La seconda è quella di scrivere un programma per muovere la griglia sullo schermo.

La terza, infine, è quella di sostituire i simboli, definiti durante la prima operazione, con l'immagine voluta (rana, carro armato, ecc.)

Occhio ai numeri!

I numeri relativi al codice macchina devono essere trascritti con estrema cura.

Una volta trasferito il codice macchina in memoria, infatti, non è possibile modificarlo come se fosse un programma BASIC. Tuttavia, i programmi BASIC qui presentati (per il carro armato e per la rana), si possono modificare a piacimento.

Inoltre, si possono alterare le forme, definite dai valori contenuti nelle frasi DATA, per impratichirsi a crearne delle nuove.

Viceversa, il programma BASIC che crea le griglie iniziali non deve esser cambiato in alcune delle sue parti e deve essere eseguito soltanto quando si è verificata l'esatta corrispondenza con l'originale.

I CARATTERI DEFINITI DALL'UTENTE

Sullo Spectrum, un simbolo grafico definito dall'utente, o più brevemente simbolo UDG (da User Definable Graphics Character), è una lettera (la A, ad esempio), che può venire rimodellata in qualcosa di nuovo.

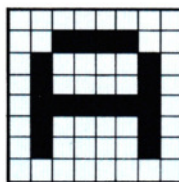



fig. 1

Ogni lettera consiste di 64 punti disposti in una matrice 8×8, (figura 1). Alcuni dei punti sono colorati, altri no: con un apposito programma, si può modificare la disposizione dei vari punti colorati.

Lo Spectrum dispone di 21 caratteri modificabili: dalla A alla U. È possibile combinare più matrici, per formare un'immagine complessa (figura 2). Sullo schermo è quindi possibile usare, simultaneamente, o due griglie di 3×3 (tre UDG rimangono inutilizzati) oppure cinque griglie di 2×2 .

LA CREAZIONE DELLA GRIGLIA

Sia per la rana, che per il carro armato, è necessaria una griglia di 3×3 caratteri UDG, ossia:

```
A B C
D E F
G H I
```

Un metodo potrebbe essere quello di usare, in BASIC, una serie di istruzioni PRINT AT:

```
PRINT AT 10,10,<ABC grafiche>
PRINT AT 11,10;<DEF grafiche>
```

e così via.

Ma un modo migliore consiste nell'impiegare il codice macchina, preparato dal programma BASIC riportato più avanti. Nel trascriverlo, si badi di usare i tasti 'chiave' dello Spectrum, senza scrivere le istruzioni per intero: CODE, ad esempio, si immette premendo contemporaneamente CAPS SHIFT, SYMBOL SHIFT e L.

```
10 IF PEEK 23733=127 THEN CLEAR
   32399:LET B=32400:LETZ=0
20 IF PEEK 23733=255 THEN CLEAR
   65199:LET B=65200:LETZ=1
30 FOR N=B TO B+129:READ A:POKE
   N,A:NEXT N
40 IF Z=1 THEN POKE 65258,178: POKE
   65259,254: POKE 65277,179: POKE
   65278,254
50 SAVE "FramePrint" CODE B,130
60 STOP
100 DATA 24,55,1,22,0,0,32,32,32,22,0,0,32,
   32,32,22,0,0,32,32
110 DATA 32,22,0,0,144,145,146,22,0,0,147,
   148,149,22,0,0,150,151,152,22
120 DATA 0,0,153,154,155,22,0,0,156,157,
   158,22,0,0,159,160,161,58,146,126
130 DATA 254,1,1,18,0,40,8,56,4,203,33,24,2,
   14,0,221,33,147,126,221
140 DATA 9,58,137,92,71,62,24,144,221,119,
   1,60,221,119,7,60,221,119,13,58
150 DATA 136,92,71,62,33,144,221,119,2,
   221,119,8,221,119,14,221,229,62,2,205
160 DATA 1,22,209,1,18,0,205,60,32,201
```

Questa routine, può sembrare di dimensioni ciclopiche, se confrontata con le sole tre istruzioni PRINT AT, ma offre due grossi vantaggi:

1 Una volta trascritta, la si può memorizzare e riutilizzare, in seguito, più volte.

2 Permette di muovere l'intera griglia, sullo schermo, con una velocità difficilmente ottenibile utilizzando un programma in solo BASIC.

Ovviamente, senza conoscere il codice macchina, il significato dei codici contenuti nelle linee DATA del programma ri-

- LA CREAZIONE DI UNA CORNICE GRAFICA MOBILE
- OTTENERE MOVIMENTI RAPIDI
- UNA RANA CHE SALTA E UN CARRO ARMATO CHE SPARA

mane oscuro. Per il momento, non possiamo che illustrarlo brevemente:
Le linee 10 e 20 individuano la zona di memoria ove depositare il codice macchina. Questa varia a seconda della memoria totale dell'apparecchio.

```
20 CLEAR 65199: LET B=65200: LET Z=1
```

La linea 30 estrae i vari codici scritti nelle frasi DATA e li trasferisce nella memoria. La linea 50 memorizza il sottoprogramma su nastro. (Il nome 'FramePrint' è puramente arbitrario e lo si può cambiare).

Le linee da 100 a 100 contengono i valori numerici che, una volta in memoria, rappresentano il programma in codice macchina.

La memorizzazione su nastro (SAVE) avviene diversamente dal solito: trascritto il programma, lo si esegue, digitando RUN. Il computer avvisa, a questo punto, di avviare il registratore e di battere 'un tasto qualsiasi' per procedere oltre.

LO SPOSTAMENTO DELLA GRIGLIA

Con il sottoprogramnma ancora in memoria (oppure riletto con una LOAD e 'lanciato' di nuovo con un RUN) si passa all'impiego vero e proprio. In primo luogo, occorre digitare un NEW, seguito da ENTER. Poi, si trascrive il seguente programma:

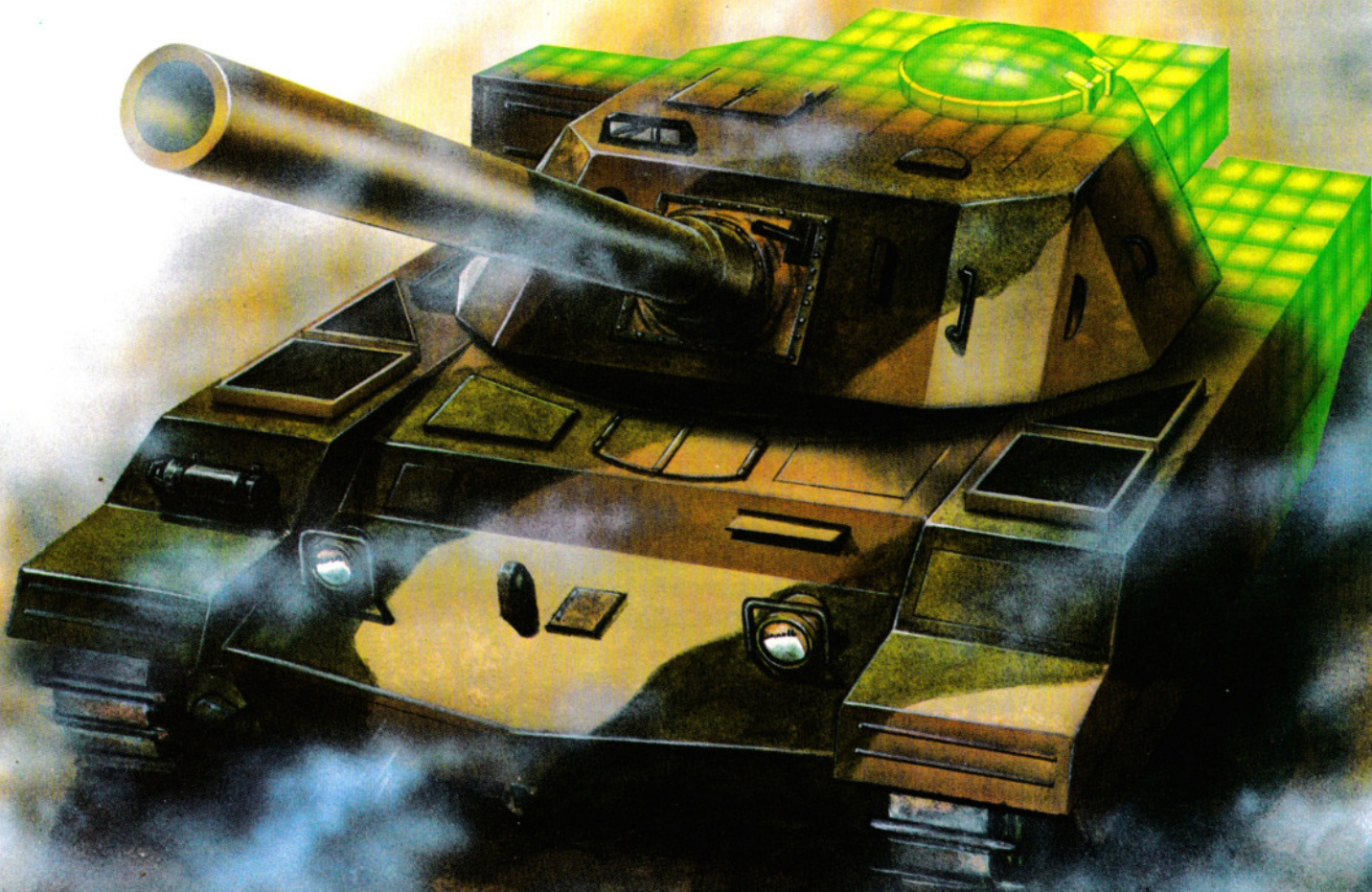
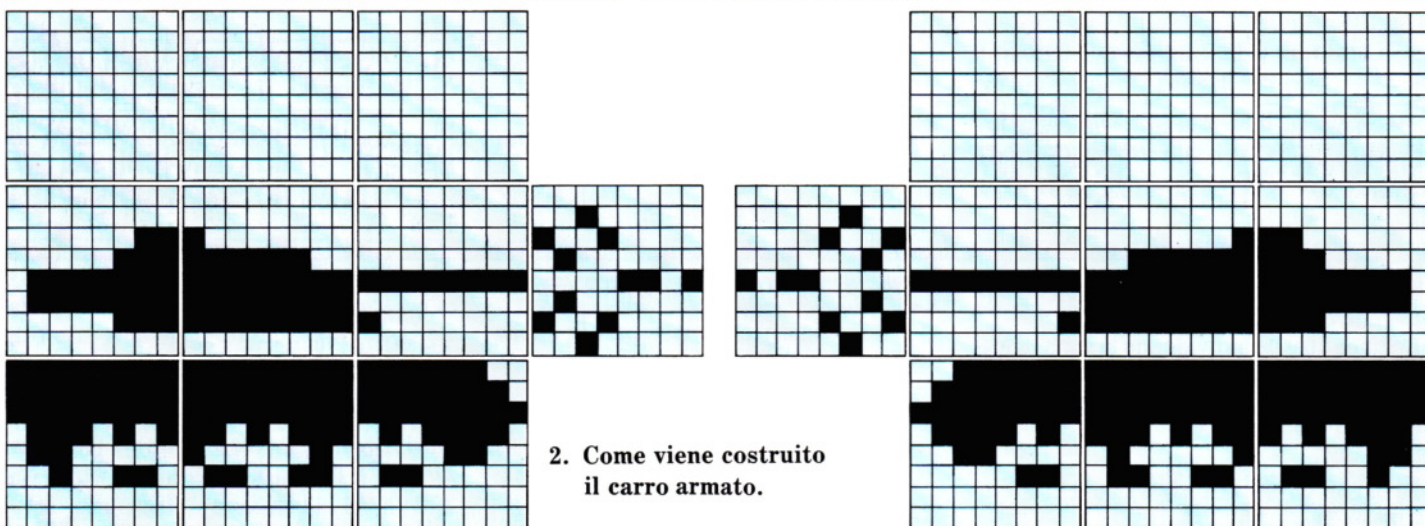
```
20 LET print=32400: LET B=32402: IF
   PEEK 23733=255 THEN LET print
   =65200: LET B=65202
90 BORDER 0: PAPER 0: INK 4: CLS
100 LET Y=8: LET X=15: LET Y1=8: LET
   X1=15: LET Z=1
110 LET A$=INKEY$
120 IF A$="z" AND X>0 THEN LET X1
   =X-1: LET Z=1
130 IF A$="x" AND X<29THEN LET X1
   =X+1: LET Z=2
140 IF A$="p" AND Y>0 THEN LET Y1
   =Y-1
150 IF A$="l" AND Y<18 THEN LET Y1
   =Y+1
170 LET X=X1: LET Y=Y1
180 PRINT AT Y,X;: POKE B,Z: RANDOMIZE
   USR print
190 GOTO 110
```

Questo programma BASIC permette di spostare l'immagine sullo schermo, in base ai comandi: P (verso l'alto), Z (verso sinistra), L (verso il basso) e X (verso destra).

Eseguendo il programma con un RUN, si nota come non si sia definita una sola griglia 3×3, ma due: la seconda, costruita con le lettere JKLMNOPQR, rappresenta l'immagine speculare della prima.

Si nota però anche un altro fatto: nello spostare l'immagine, essa lascia dietro di sé una scia, indesiderata, di caratteri. Per eliminare l'inconveniente, basta aggiungere:

```
160 PRINT AT Y,X;: POKE B, 0: RANDOMIZE
   USR print
```

Ciò genera una griglia 3×3 piena di spazi, che cancella l'immagine preesistente nella posizione X,Y.

2. Come viene costruito il carro armato.

CREAZIONE DEL CARRO ARMATO

Per ottenere l'immagine di un carro armato, usando la griglia precedente, è sufficiente apportare le seguenti modifiche al programma, che definiscono un carro armato rivolto a sinistra ed uno a destra:

```
10 FOR N=USR "a" TO USR "r"+7: READ
   A: POKE N,A: NEXT N
1000 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1010 DATA 0,0,0,0,0,0,0,0,0,0,0,0,255,0,1,0
1020 DATA 0,0,1,63,255,255,255,0,0,0,192,
   224,254,254,224,0
1030 DATA 63,127,255,122,48,6,0,0,255,255,
   255,235,65,102,0,0
1040 DATA 255,255,255,174,6,100,0,0
1050 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1060 DATA 0,0,0,0,0,0,0,0,0,0,3,7,127,127,7,0
1070 DATA 0,0,128,252,255,255,255,0,0,0,0,
   0,255,0,128,0
1080 DATA 255,255,255,117,96,38,0,0,255,
   255,255,215,130,102,0,0
1090 DATA 252,254,255,94,12,96,0,0
```

Per 'armare' il carro armato, occorrono altri due UDG. Premere il tasto BREAK ed aggiungere:

```
10 FOR N=USR "a" TO USR "t"+7: READ
   A: POKE N,A: NEXT N
115 IF INKEY$="□" THEN GOTO 200
200 IF Z=2 THEN GOTO 300
210 FOR N=X-1 TO 0 STEP-1
220 PRINT INK 5;AT Y+1,N; CHR$ 162
230 PAUSE 1
240 PRINT AT Y+1,N;"□"
250 NEXT N
260 GO TO 110
300 FOR N=X+3 TO 31
310 PRINT INK 5;AT Y+1,N; CHR$ 163
320 PAUSE 1
330 PRINT AT Y+1,N;"□"
340 NEXT N
350 GO TO 110
1100 DATA 0,4,9,2,176,2,9,4,0,32,144,64,13,
   64,144,32
```

Fatto ripartire il programma con un RUN, si possono adesso 'sparare' proiettili, con la semplice pressione del tasto SPACE.

CREAZIONE DELLA RANA

Per definire l'immagine della rana occorre, in primo luogo, cancellare quella del carro armato e ciò si ottiene digitando il comando NEW, seguito da ENTER.

Questa operazione provoca la perdita sia della precedente definizione che del relativo programma, ma il sottoprogramma in codice macchina è ancora presente in memoria purché, nel frattempo, non si sia spento l'apparecchio!

In tal caso, digitare CLEAR 32399 (sullo Spectrum da 16 Kbyte) o CLEAR 65199 (su quello da 48 Kbyte). Questa operazione serve ad accantonare uno spazio sufficiente di memoria per il sottoprogramma in codice macchina, senza il rischio che esso venga distrutto dalla presenza del BASIC. A questo punto, si digiti LOAD "" CODE e si avvii il registratore.

La definizione dell'immagine della rana si ottiene con le seguenti linee di programma:

```
30 RESTORE 5000: FOR N=USR "a" TO USR
   "r" +7: READ A: POKE N,A: NEXT N
5000 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
5010 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1
5020 DATA 0,0,0,0,0,128,192,176,0,0,0,0,0,0,
   0,0
5030 DATA 4,15,31,63,127,254,248,127,96,
   240,224,192,64,32,156,192
5040 DATA 0,0,0,0,0,0,0,0
5050 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,1,3,7
5060 DATA 8,28,27,70,255,254,252,249,0,0,0,
   0,0,0,0,0
5070 DATA 7,7,15,30,54,38,70,70,250,196,0,
   0,0,0,0,0
5080 DATA 0,0,1,1,3,6,2,0,140,144,16,32,32,
   48,32,0
5090 DATA 0,0,0,0,0,0,0,0
```

COME MUOVERE LA RANA

Per spostare l'immagine della rana sullo schermo, immettere:

```
10 BORDER 0: PAPER 0: INK 4: BRIGHT 1:
   CLS
20 LET P=32400: IF PEEK 23733=255
   THEN LET P=65200
100 PRINT AT 10,0;:RANDOMIZE USR P: IF
   INKEY$=" " THEN GOTO 100
110 RESTORE 1000:FOR F=1 TO 5
120 READ A,B,C: POKE P+2,A: PRINT AT
   B,C;: RANDOMIZE USR P
130 PAUSE 2: CLS: NEXT F
150 PRINT AT 10,12;: RANDOMIZE USR P: IF
   INKEY$=" " THEN GOTO 150
200 FOR F=1 TO 5
210 READ A,B,C: POKE P+2,A: PRINT AT
   B,C;: RANDOMIZE USR P
220 PAUSE 2: CLS: NEXT F
```

```
230 GOTO 100
1000 DATA 1,10,0,2,7,3,2,5,6,2,7,9,1,10,12,1,
    10,12,2,7,15,2,5,18,2,7,21,1,10,24
```

Per costruire le immagini grafiche riportate nelle figure 2 e 3, sono necessarie tre operazioni.

La prima è quella di definire, nella memoria del computer, una cornice o 'griglia', onde stabilire le dimensioni dell'immagine grafica. Inizialmente, la definizione avviene mediante una serie di *simboli grafici definiti dall'utente* (vedere al paragrafo seguente).

La seconda è quella di scrivere un programma per muovere la griglia sullo schermo.

La terza, infine, è quella di sostituire i simboli, definiti durante la prima operazione, con l'immagine voluta (rana, carro armato, ecc.)

I CARATTERI DEFINITI DALL'UTENTE

Sull'Electron e sul BBC Modello B, un carattere grafico definito dall'utente, o più brevemente un carattere UDG (da User Definable Graphics), è paragonabile a una 'griglia' nella quale costruire una figura.

Ogni 'griglia' è composta da 64 punti, disposti in una matrice 8×8. Inizialmente, i punti hanno lo stesso colore dello sfondo e risultano, pertanto, invisibili. Ma, modificando il colore di alcuni di essi, si possono creare nuove figure.

Normalmente, si possono creare fino a 32 caratteri UDG, che si possono affiancare l'uno all'altro come è mostrato nella figura 2.

Sullo schermo, si possono visualizzare contemporaneamente tre griglie di 3×3 caratteri UDG (cinque UDG non vengono utilizzati) oppure otto griglie di 2×2, o, addirittura, una sola griglia di 8×4.

LA CREAZIONE DELLA GRIGLIA

Sia per la rana che per il carro armato, qui descritti, occorre una griglia di 3×3 caratteri UDG, i cui valori sono:

224	225	226
227	228	229
230	231	232

Si potrebbe usare un semplice programma in BASIC ricorrendo all'istruzione PRINT TAB:

```
PRINT TAB (10,10);CHR$(224);CHR$(225);
    CHR$(2266
PRINT TAB (10,11);CHR$(227);CHR$(228);
    CHR$(229)... e così via.
```

Ma un metodo migliore, anche se più laborioso, consiste nell'usare un programma in BASIC per generare un sottoprogramma in codice macchina. Questo sistema offre due vantaggi: una volta creato, il sottopogramma può essere riutilizzato in seguito e, soprattutto, la velocità d'esecuzione è notevolmente superiore a quella di un programma in BASIC. Prima di trascrivere il seguente programma, digitare il comando NEW:

```
10 FOR T=&D00 TO &D58
20 READ A:?T=A
30 NEXT T
40 *SAVE "FramePrint" 0D00 +59
50 DATA 24,169,224,202,48,47,240,7,202,
    240,2,105,9,105,9,162,3,160,3,32,238,255,
    24,105,1,136,208,247,202,240,21,72,169
60 DATA 10,32,238,255,169,8,32,238,255,32,
    238,255,32,238,255,104,76,17,&0D,96,
    169,32,162,3,160,3,32,238,255,136,208,
    250
70 DATA 202,240,240,169,10,32,238,255,
    169,8,32,238,255,32,238,255,32,238,255,
    169,32,76,57,&0D
```

Nel caso si possieda un'unità a dischetti, apportare le seguenti modifiche. Cambiare la linea 10 in: 10 FOR T=&A00 TO &A58. Nella linea 40 sostituire 0D00 con 0A00. Nelle linee 60 e 70 sostituire &0D con &0A.

Adesso si esegua il programma mediante un RUN. Il computer chiederà di avviare il registratore (se usato) e di premere il tasto RETURN.

Eseguite queste operazioni, il sottoprogramma è memorizzato e può essere riutilizzato in futuro.

NON USARE IL TASTO BREAK

Attenzione: la pressione del tasto BREAK provoca la perdita dei programmi temporaneamente conservati in memoria. Evitarne l'uso, quindi, quando il programma appena presentato è in memoria e s'intende usarlo.

Non conoscendo il codice macchina, la serie di numeri nelle frasi DATA del programma risulta incomprensibile. Per adesso basti sapere che:
Le linee da 10 a 30 estraggono i valori contenuti nelle frasi DATA e li depositano nella memoria del computer.
Nella linea 40 è contenuto il comando che memorizza il sottoprogramma su nastro. Il nome 'FramePrint' può essere modificato a piacimento.

LO SPOSTAMENTO DELLA GRIGLIA

Adesso che abbiamo memorizzato una coppia del sottoprogramma, possiamo tranquillamente provarlo.

In primo luogo, si usi il comando NEW per rimuovere il vecchio programma BASIC (quello in codice macchina non viene leso da questo comando). Successivamente, per ottenere il movimento della griglia sullo schermo, si trascriva il seguente programma:

```
10 MODE 1
20 VDU 23;8202;0;0;0;
40 X=20:Y=20:X1=20:Y1=20:Z=1
50 A$=GET$
60 IF A$="Z" AND X>0 THEN X1=X
    -1:Z=1
70 IF A$="X" AND X<37 THEN X1=X
    +1:Z=2
80 IF A$="L" AND Y<29 THEN Y1=Y+1
90 IF A$="P" AND Y>0 THEN Y1=Y-1
120 X=X1:Y=Y1
130 VDU 31,X,Y:X%=Z:CALL &D00
140 GOTO 50
```

(Se si usano unità a dischetti, alla linea 130 sostituire &D00 con &A00)

Impartendo un RUN, si può adesso spostare la griglia sullo schermo, usando i comandi: P (verso l'alto), Z (verso sinistra), L (verso il basso) e X (verso destra).

Non è detto, però, che l'esecuzione del programma abbia alcun effetto pratico: non è stato ancora definito, infatti, alcun carattere UDG da usare nella griglia!

LA CREAZIONE DEL CARRO ARMATO

Si interrompa, perciò, l'esecuzione del programma precedente premendo il tasto ESCAPE e si trascriva il seguente programma, che serve a definire l'immagine del carro armato (figura 2):

```
30 GOSUB 260
260 VDU 23,224,0,0,0,0,0,0,0,0,23,225,0,0,0,0,
    0,0,0,0
2700 VDU 23,226,0,0,0,0,0,0,0,0,23,227,0,0,0,
    0,255,0,1,0
280 VDU 23,228,0,0,1,63,255,255,255,0,23,
```

```
    229,0,0,192,224,254,254,224,0
290 VDU 23,230,63,127,255,122,48,6,0,0,23,
    231,255,255,255,235,65,102,0,0
300 VDU 23,232,255,255,255,174,6,100,0,0
310 VDU 23,233,0,0,0,0,0,0,0,0,23,234,0,0,0,0,
    0,0,0,0
320 VDU 23,235,0,0,0,0,0,0,0,0,23,236,0,0,3,7,
    127,127,7,0
330 VDU 23,237,0,0,128,252,255,255,255,0,
    23,238,0,0,0,0,255,0,128,0
340 VDU 23,239,255,255,255,117,96,38,0,0,
    23,240,255,255,255,215,130,102,0,0
350 VDU 23,241,252,254,255,94,12,96,0,0
380 RETURN
```

Eseguendo il programma, si noterà che il carro armato lascia una scia dietro di sé: per eliminarla, si aggiunga:

```
110 VDU 31,X,Y:X%=0: CALL &D00
```

(con i dischetti sostituire &D00 con &A00). Questa linea di programma crea una griglia 3×3 piena di spazi, che cancella ciò che si trova in posizione X,Y.
Dopo aver sperimentato la creazione del carro armato, si possono creare anche altre forme, purché non si esca dalla griglia 3×3.

Sono stati usati soltanto 18 dei 32 caratteri UDG disponibili: possiamo far 'sparare' il carro armato, definendo una forma appropriata. Per far ciò si *aggiungano* al programma precedente le seguenti linee:

```
100 IF A$="□" THEN GOTO 150
150 IF Z=2 THEN GOTO 210
160 FOR T=X-1 TO 0 STEP-1
170 VDU 31,T,Y+1,242,8
180 A$=INKEY$(5):VDU32
190 NEXT T
200 GOTO 50
210 FOR T=X+3 TO 39
220 VDU 31,T,Y+1,243,8
230 A$=INKEY$(5):VDU 32
240 NEXT T
250 GOTO 50
360 VDU 23,242,0,4,9,2,176,2,9,4
370 VDU 23,243,0,32,144,64,13,64,144,32
```

La pressione del tasto 'spazio' provoca il 'fuoco'.

LA CREAZIONE DELLA RANA

Per creare l'immagine della rana, si procede in modo analogo. Prima di tutto si usa il comando NEW, che cancella la definizione del carro armato, lasciando, però, il sottoprogramma ancora attivo. (Questo viene cancellato soltanto se si spegne il computer o se si preme il tasto BREAK).

Se si è appena acceso il computer, occorre rileggere il programma memorizzato su nastro, usando il comando LOAD "".

Adesso siamo pronti per immettere la definizione della rana:

```
30 VDU 23,224,0,0,0,0,0,0,0,0,23,225,0,0,0,0,
   0,0,0,0,23,226,0,0,0,0,0,0,0,0
40 VDU 23,227,0,0,0,0,0,0,1,1,23,228,0,0,0,0,
   0,128,192,176
50 VDU 23,229,0,0,0,0,0,0,0,0,23,230,4,15,31,
   63,127,254,248,127
60 VDU 23,231,96,240,224,192,64,32,156,
   192,23,232,0,0,0,0,0,0,0,0
70 VDU 23,233,0,0,0,0,0,0,0,0,23,234,0,0,0,0,
   0,1,3,7
80 VDU 23,235,8,28,27,70,255,254,252,249,
   23,236,0,0,0,0,0,0,0,0
90 VDU 23,237,7,7,15,30,62,54,70,70,23,238,
   250,196,0,0,0,0,0,0
100 VDU 23,239,0,0,1,1,3,6,2,0,23,240,140,
    144,16,32,32,48,32,0
110 VDU 23,241,0,0,0,0,0,0,0,0
```

COME MUOVERE LA RANA

Queste poche righe di programma faranno 'saltare' la rana ad ogni pressione della barra spaziatrice:

```
10 MODE 1
20 VDU 23;8202;0;0;0
120 X=0:Y=20:DY=0
130 VDU 31,X,Y
140 X%=1:CALL &D00
150 IF GET$=" " THEN GOSUB 180
160 IF X>35 THEN VDU 31,X,Y:X%
    =0:CALL &D00:GOTO 120
170 GOTO 150
180 RESTORE:FOR T=1 TO 4
190 VDU 31,X,Y-DY
200 X%=0:CALL &D00
210 READ F,DY
220 X=X+3
230 VDU 31,X,Y-DY
240 X%=F:CALL &D00
250 A$=INKEY$(5)
260 NEXT T
270 RETURN
280 DATA 2,3,2,5,2,3,1,0
```

(Con i dischetti, apportare i seguenti cambiamenti: sostituire &D00 con &A00 nelle linee 140, 160, 200 e 240). Avviare il programma. I valori nella frase DATA stabiliscono l'altezza dei salti.

Per produrre le immagini riportate nelle figure 1 e 2, sono necessarie tre operazioni.

La prima consiste nel definire, nella memoria del computer, una griglia di dimensioni adeguate. Questa griglia è composta da *caratteri grafici definiti dall'utente*, o più brevemente UDG (da User Definable Graphics), che inizialmente, sul Dragon, non sono visibili.

La seconda consiste nel fornire le specifiche dell'immagine da riprodurre, mediante frasi DATA.

La terza, infine, consiste nello scrivere un programma che consenta di visualizzare le immagine create e di farle muovere sullo schermo.

I CARATTERI UDG DEL DRAGON

Un carattere UDG è paragonabile ad una griglia, nella quale viene definita una figura. La griglia è composta da 64 punti, disposti in una matrice 8×8. Ciascun punto, in PMODE 4,1, può essere bianco o nero (la grafica PMODE 4,1 è la più efficace per queste operazioni).

Raggruppando più caratteri UDG, si possono comporre immagini complesse, chiamate anche *cornici*. Diversamente da altri microcomputer, il Dragon non possiede caratteri UDG predefiniti, ma è possibile simularli, usando opportunatamente la memoria dell'apparecchio.

Teoricamente, si possono creare centinaia di queste cornici, ma il loro impiego più o meno simultaneo comporterebbe un notevole sforzo di programmazione. In genere, ne bastano quattro o cinque soltanto.

LA CREAZIONE DELLA GRIGLIA

Per riprodurre la rana (figura 3) o il carro armato (figura 2) occorre prima definire una griglia di 3×3 caratteri UDG, come quella riprodotta in figura 1, ma vuota!

Disporremo così di una griglia ideale di 24×24 punti, da poter muovere sullo schermo.

Se usassimo il BASIC del Dragon, occorrerebbe definire le caratteristiche di ciascuno dei 576 punti, mediante l'istruzione PSET. In codice macchina ciò è molto più facile e veloce.

Si trascriva il seguente programma:

```
10 CLEAR 200,32000
20 FOR I=32000 TO 32110
30 READ N
40 POKE I,N
50 NEXT
60 CLS
90 PRINT "PREMERE UN TASTO QUALSIASI
   PER SALVARE IL CODICE MACCHINA"
100 B$=INKEY$
110 IF B$="" THEN 100
120 CSAVEM"FRAMEPRN",32000,32110,
   32000
130 DATA 190,127,188,134,3,183,125,111,
   183,125,112,134,8,183,125,113
140 DATA 182,125,250,39,50,206,126,44,74,
   198,72,61,51,203,166,192
150 DATA 167,132,48,136,32,122,125,113,
   38,244,134,8,183,125,113,48
160 DATA 137,255,1,122,125,111,38,
   230,134,3,183,125,111,48,137,0
170 DATA 253,122,125,112,38,216,
   57,95,231,132,48,136,32,122,125,113
180 DATA 38,246,134,8,183,125,113,48,137,
   255,1,122,125,111,38,232
190 DATA 134,3,183,125,111,48,137,0,253,
   122,125,112,38,218,57
```

Adesso siamo in grado di memorizzare il sottoprogramma in codice macchina, onde riutilizzarlo in futuro. Per questa operazione, naturalmente, occorre approntare il registratore.

La versione in codice macchina, rispetto all'equivalente in BASIC, è notevolmente più rapida e consuma meno memoria.

Senza una conoscenza del codice macchina 68Ø9, i numeri contenuti nelle frasi DATA non hanno alcun significato. Basti sapere, per adesso, che:

La linea 1Ø 'accantona' una zona di memoria riservata al codice macchina, proteggendola, al tempo stesso, da interefeerenze da parte di successivi programmi BASIC.

Le linee da 9Ø a 12Ø servono a memorizzare il sottoprogramma in codice macchina su nastro.

LA COSTRUZIONE DEL CARRO ARMATO

Per l'immagine del carro armato definiamo due griglie: in una il carro è rivolto a sinistra, nell'altra a destra.

Digitare NEW, per liberare la memoria dal programma BASIC precedente (ma non di quello in codice macchina), poi trascrivere:

```
10 CLEAR 200,32000
20 FOR I=32300 TO 32443
30 READ N
40 POKE I,N
50 NEXT
```

```
60 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0
70 DATA 0,0,3,7,127,127,7,0,0,0,128,252,255,
   255,255,0
80 DATA 0,0,0,0,255,0,128,0,255,255,255,
   117,96,38,0,0
90 DATA 255,255,255,215,130,102,0,0,252,
   254,255,94,12,96,0,0
100 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0
110 DATA 0,0,0,0,255,0,1,0,0,0,1,63,255,255,
   255,0
120 DATA 0,0,192,224,254,254,224,0,63,127,
   255,122,48,6,0,0
130 DATA 255,255,255,235,235,65,102,0,0,
   255,255,255,174,6,100,0,0
```

Questo programma definisce la forma del carro armato, ma ancora non la visualizza.

VISUALIZZAZIONE DEL CARRO ARMATO

Il seguente programma in BASIC visualizza e permette di muovere il carro armato sullo schermo. Non si usi il comando NEW, che cancellerebbe le frasi DATA precedentemente immesse.

```
5 PCLEAR 5
170 PMODE 4,1
180 PCLS
290 PCLS
300 SCREEN 1,1
310 T=1
320 TP=3500
330 POKE 32700, INT(TP/256)
340 POKE 32701, TP-256*INT(TP/256)
350 POKE 32250, T
360 EXEC 32000
370 LP=TP
380 IF PEEK (338)=239 THEN TP=TP
   -32:GOTO 440
390 IF PEEK (342)=247 THEN TP=TP
   +32:GOTO 440
400 IF PEEK (340)=223 THEN TP=TP
   -1:T=2:GOTO 440
410 IF PEEK (338)=223 THEN TP=TP
   +1:T=1:GOTO 440
430 GOTO 380
440 IF TP<1536 OR TP>6941 THEN TP
   =LP
470 GOTO 330
```

Adesso, con il comando RUN, si esegue il programma. Per muovere il carro armato usare i tasti: P (verso l'alto), Z (verso sinistra), L (verso il basso) ed X (verso destra).

Si noterà che il carro lascia una scia dietro di sé. Per eliminarla, si aggiungano le seguenti linee, digitando un RUN:

```
450 POKE 32250,0
460 EXEC 32000
```

Con queste istruzioni si ripulisce lo schermo mediante una griglia piena di spazi.

IL CARRO ARMATO SPARA

Per far sì che il carro 'spari' dei proiettili, si possono aggiungere le seguenti linee di BASIC.

La barra spaziatrice, sulla tastiera, serve da 'pulsante di fuoco'.

```
160 DIM A(2), B(2), C(2)
190 FOR I=1536 TO 1760 STEP 32
200 READ N
210 POKE I, N
220 NEXT
230 GET (0,0)-(7,7),A
240 FOR I=1536 TO 1760 STEP 32
250 READ N
260 POKE I, N:NEXT
280 GET (0,0)-(7,7),B
420 IF PEEK (345)=223 THEN GOSUB 500
480 DATA 0,32,144,64,13,64,144,32
490 DATA 0,4,9,2,176,2,9,4
500 IF T=2 THEN 560
510 YP=INT ((TP-1536)/32)+8
520 XP=8* (TP-1533-(YP-8)*32)
530 IF XP>255 THEN 620
540 PUT (XP,YP)-(XP+7,YP+7),A
550 GOTO 600
560 YP=INT ((TP-1536)/32)+8
570 XP=8* (TP-1537-(YP-8)*32)
580 IF XP<0 THEN 620
590 PUT (XP,YP)-(XP+7,YP+7),B
600 FOR I=1 TO 100:NEXT
610 PUT (XP,YP)-(XP+7,YP+7),C
620 RETURN
```

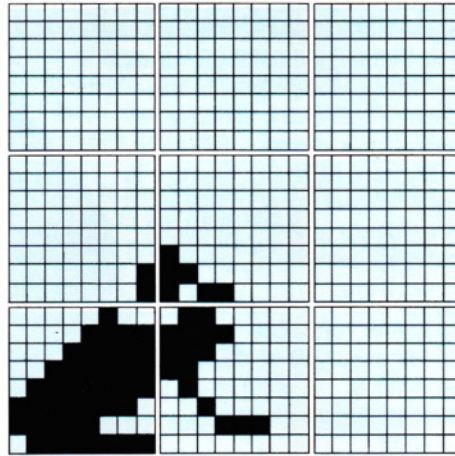
LA CREAZIONE DELLA RANA

La figura 3 mostra le griglie necessarie per definire l'immagine della rana 'a riposo' e durante un salto.

Se il computer non è stato spento nel frattempo, il sottoprogramma in codice macchina, è ancora presente in memoria. Altrimenti, digitare CLEAR 200,32000 (per riservare una zona di memoria al sottoprogramma) e quindi CLOADM, avviando il registratore (per leggere il codice macchina in memoria).

Le linee BASIC che seguono servono a definire l'immagine della rana. Si usi il comando NEW per rimuovere eventuali programmi BASIC precedenti, poi si trascriva:

```
10 CLEAR 200,32000
20 FOR I=32300 TO 32443
30 READ N
40 POKE I, N:NEXT
60 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0
70 DATA 0,0,0,0,0,0,1,1,0,0,0,0,0,128,192,176
80 DATA 0,0,0,0,0,0,0,0,4,15,31,63,127,254,
   248,127
90 DATA 96,240,224,192,64,32,156,192,0,0,0,
   0,0,0,0,0
```

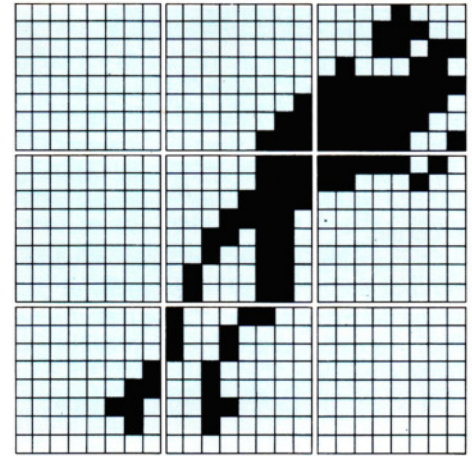
3. La grafica della rana. Le griglie sono di 8×8 pixel.

```
100 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,3,7,8,28,
    27,70,255,254,252,249
110 DATA 0,0,0,0,0,0,0,0,0,7,7,15,30,62,54,70,
    70
120 DATA 250,196,0,0,0,0,0,0,0,0,0,1,1,3,6,2,0
130 DATA 140,144,16,32,32,48,32,0,0,0,0,0,0,
    0,0,0
```

I MOVIMENTI DELLA RANA

Il seguente programma serve a far 'saltare' la rana. Si impartisca un NEW e si immetta:

```
10 PCLEAR 5
20 PMODE 4,1
30 PCLS
40 SCREEN 1,1
50 FP=4487
60 POKE 32700,17
70 POKE 32701,135
80 POKE 32250,1
90 EXEC 32000
100 IF PEEK(345)<>223 THEN 100
110 RESTORE
120 FOR I=1 TO 5
130 READ F,FD
140 POKE 32700,INT (FP/256)
150 POKE 32701,FP-256*INT (FP/256)
160 POKE 32250,F
170 EXEC 32000
180 FOR J=0 TO 70:NEXT
190 POKE 32250,0
200 EXEC 32000
210 FP=FP+FD
220 NEXT
230 GOTO 80
240 DATA 1,-287,2,-253,2,258,2,291,1,0
```

I valori contenuti nell'ultima frase DATA stabiliscono le caratteristiche del salto. Per far saltare la rana, agire sulla barra spaziatrice della tastiera.

Il miglior modo per riprodurre le immagini delle figure 2 e 3 è quello di usare una speciale funzione grafica del Commodore 64: gli *sprite*, una sorta di caratteri UDG, ma di grandi dimensioni.

LA CREAZIONE DEL CARRO ARMATO

Il seguente programma BASIC è composto da tre parti: una sequenza di preparazione, una di controllo (per regolare i movimenti) e una di valori numerici che definiscono l'immagine:

```
10 FOR M=252 TO 253:FOR I=
   64*M TO 64*M+63:POKE I,0:NEXT
20 FOR I=64*M+21 TO 64*M+56:
   READ A:POKE I,A:NEXT:NEXT
60 SC=53248:X3=24:X=24:Y3=157:Y=
   157
70 POKE 2043,252:POKE 2042,254:POKE SC
   +23,0:POKE SC+29,0:POKE SC
   +42,12:POKE SC+27,0
90 J=0:R=0:POKE 650,128:PRINT
   "[CLR]":POKE SC+21,8
100 GET A$:IF A$="" THEN 180
110 IF A$="Z"AND X3>24 THEN X3=X3
    -12
120 IF A$="X"AND X3<310 THEN X3=
    X3+12
130 IF A$="P"AND Y3>50 THEN Y3=Y3
    -12
140 IF A$="L"AND Y3<220 THEN Y3=
    Y3+12
180 IF X<X3 THEN X=X+3:POKE
    2043,253
190 IF X>X3 THEN X=X-3:POKE
    2043,252
200 IF Y<Y3 THEN Y=Y+3
210 IF Y>Y3 THEN Y=Y-3
220 XA=INT(X/256):XB=X-XA*256
230 POKE SC+6,XB:POKE SC+7,Y
240 POKE SC+16,4*(1 AND JA)+
    8* (1 AND XA)
250 GOTO100
1000 DATA 0,1,192,0,63,224,255,255,254,0,
     255,254,1,255,224,0,0,0
```

```
1010 DATA 63,255,255,127,255,255,255,255,
    255,122,235,174,48,65,6
1020 DATA 6,102,100
1030 DATA 3,128,0,7,252,0,127,255,255,127,
    255,0,7,255,128,0,0,0
1040 DATA 255,255,252,255,255,254,255,
    255,255,117,215,94,96,130,12
1050 DATA 38, 102, 96
```

Trascritto il programma, si usi il comando RUN per eseguirlo. Sullo schermo appare un carro armato. Per muoverlo, usare i tasti Z (verso sinistra), X (verso destra) P (verso l'alto) ed L (verso il basso). Per uscire dal programma, infine, premere il tasto RUN/STOP. Si noti come l'immagine del carro giri su se stessa quando si cambia direzione di marcia.

UN PO' D'AZIONE

Fin qui abbiamo usato un solo sprite, ma ne possiamo adoperare un'altro per definire l'immagine di un proiettile. Aggiungendo le seguenti linee di programma, alla pressione della barra spaziatrice sulla tastiera, il carro armato 'spara' un proiettile attraverso lo schermo:

```
30 FOR M=254 TO 255:FOR I=64*M TO
    64*M+62:POKE I,0:NEXT
40 M2=0:IF M=255 THEN M2=2
50 FOR I=64*M+18 TO 64*M+36
    STEP3:READA:POKE I+M2,A:NEXT:NEXT
80 POKE SC+23,0:POKE SC+29,0:POKE SC
    +41,1
150 IF A$< >"□" THEN 180
160 RT=(PEEK(2043)=253):X2=-360*
    RT:Y2=Y:J=X-24*RT-11:
    POKE 2042,254-RT
170 POKE SC+21,12:GOSUB 260
260 FOR T=J TO X2-4 STEP(RT*2+1)*-
    20
270 JA=INT(T/256):JB=T-JA*256
280 POKE SC+16,4*(1 AND JA)+8*(1 AND
    XA)
290 POKE SC+4,JB:POKE SC+5,Y
300 FOR P=1 TO 30:NEXT:NEXT:RETURN
1500 DATA 4,9,2,176,2,9,4
1510 DATA 32,144,64,13,64,144,32
```

Per memorizzare il programma su nastro, digitare POKE 53269,0 (per cancellare dallo schermo gli sprite) e quindi si usi il normale comando SAVE del BASIC. Attenzione: omettendo di cancellare gli sprite, prima del SAVE, il programma viene danneggiato!

LA CREAZIONE DELLA RANA

Per far rientrare l'immagine della rana in uno sprite, occorre ridurre, anche se di poco, le dimensioni della griglia (figura 3): se eliminiamo 3 delle 24 righe, l'unica differenza percettibile è soltanto un piccolo accorciamento delle zampe.

Si spenga e si riaccenda il computer (o più semplicemente si digiti NEW RETURN. Il programma è:

```
10 FOR M=252 TO 253:FOR I=64*M TO
    64*M+29:POKE I,0:NEXT
20 FOR I=64*M+30+30* (M=253) TO
    64*M+62:READ A:POKE I,A:NEXT:NEXT
30 PRINT "♥"
40 SC=53248:X=24:Y=155:POKE
    2043,252:POKE SC+7,Y:POKE SC
    +6,X:POKE SC+16,0
50 POKE SC+23,0:POKE SC+29,0:POKE SC
    +27,0:POKE SC+42,5:POKE SC+21,8
60 SC=53248:X=24:Y=155:POKE 2043,
    252:POKE SC+7,Y:POKE SC+6,X:
    POKE SC+16,0:YR=0
70 GET A$
80 IF Y>155 THEN POKE 2043,252
90 IF PEEK (2043)=253 THEN 120
100 IF A$< >"□"THEN 70
110 POKE 2043,253:YR=7,5
120 Y=Y-YR:YR=YR-.7
130 X=X+4:IF X=296 THEN POKE
    2043,252:FOR T=1 TO 200:NEXT:GOTO
    60
140 XA=INT(X/256):XB=X-XA*256
150 POKE SC+6,XB:POKE SC+7,Y:POKE SC
    +16,(1 AND XA)*8:GOTO70
1000 DATA 0,128,0,1,192,0,1,176,0
1010 DATA 4,96,0,15,240,0,31,224,0,63,192,0
1020 DATA 127,64,0,254,32,0,248,156,0,127,
    192,0
1030 DATA 0,0,28,0,0,27,0,0,70,0,0,255,0,1,
    254,0,3,252,0,7,249
1040 DATA 0,7,250,0,7,196,0,15,0,0,30,0,0,
    62,0,0,54,0,0,70,0
1050 DATA 0,140,0,0,144,0,1,16,0,1,16,0,1,
    32,0,3,32,0,6,48,0,2,32,0
```

D+R

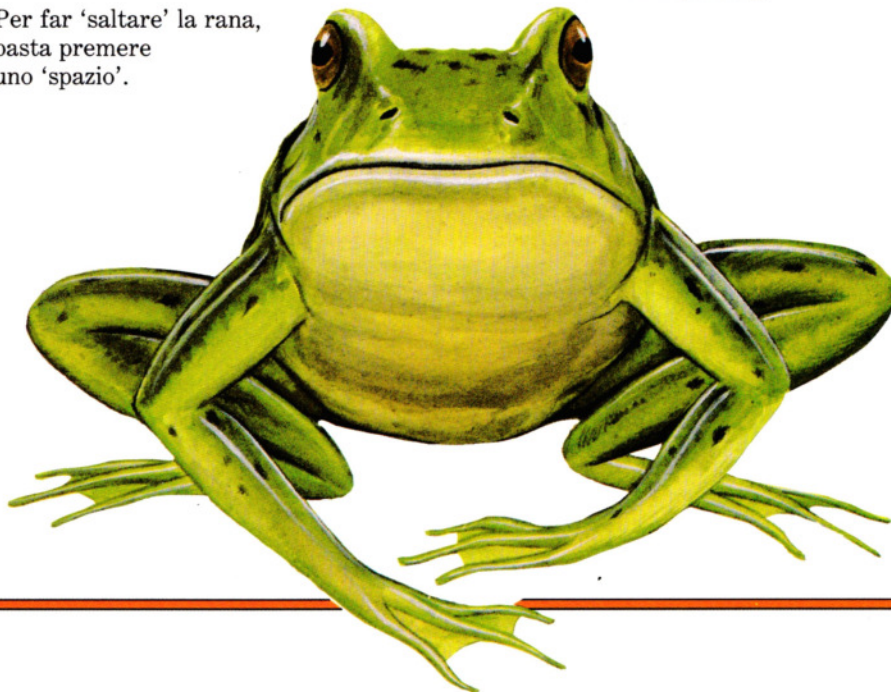
Cos'è uno sprite?

Uno *sprite* è una specie di carattere grafico definito dall'utente (più brevemente UDG, da User Definable Graphics), o anche di *codice di oggetto mobile* (più brevemente MOB, da Movable Object Code). Il Commodore 64, grazie a un particolare sistema grafico, permette di definire immagini mobili mediante gli sprite. Contrariamente agli UDG, gli sprite facilitano l'animazione delle immagini e sono programmabili punto per punto. La griglia di uno sprite è composta da 24×21 pixel, quella di un normale carattere, invece, è di 8×8 pixel.

Generalmente, sullo schermo si possono adoperare, contemporaneamente, fino a otto sprite, ma questo limite può essere superato con artifici di programmazione. Gli sprite si possono anche disporre in modo da creare un effetto tridimensionale.

Altri vantaggi degli sprite sono: la facilità con la quale si raddoppiano le loro dimensioni, la possibilità di rilevare collisioni con altri spirte e quella di creare forme multicolori.

La programmazione relativa alla grafica 'a sprite' è molto più facile e di maggior effetto che non quella relativa ai caratteri UDG, come vedremo in seguito.

Per far 'saltare' la rana, basta premere uno 'spazio'.

L'ARTE DI USARE I CICLI FOR ... NEXT

Un ciclo FOR ... NEXT serve per far contare il computer fino a un numero prefissato: questa funzione, apparentemente semplice, ha un'infinità di applicazioni

I cicli FOR ... NEXT sollevano il programmatore da molte fastidiose ripetizioni, all'interno dei programmi.

Quando, ad esempio, vogliamo che il computer ripeta una serie di operazioni un numero prefissato di volte, si usa un ciclo FOR ... NEXT.

Come vedremo, i cicli FOR ... NEXT si possono anche usare per 'dipingere' col computer, ma sono anche utili nei giochi e nei programmi applicativi.

COS'È UN CICLO FOR ... NEXT

In BASIC, un ciclo FOR ... NEXT è una particolare struttura per far ripetere al computer più volte una stessa operazione.

Supponiamo, per esempio, di voler trovare le radici quadrate di tutti i numeri compresi tra 1 e 100. Potremmo scrivere:

```
PRINT SQR(1)
PRINT SQR(2)
PRINT SQR(3)
```

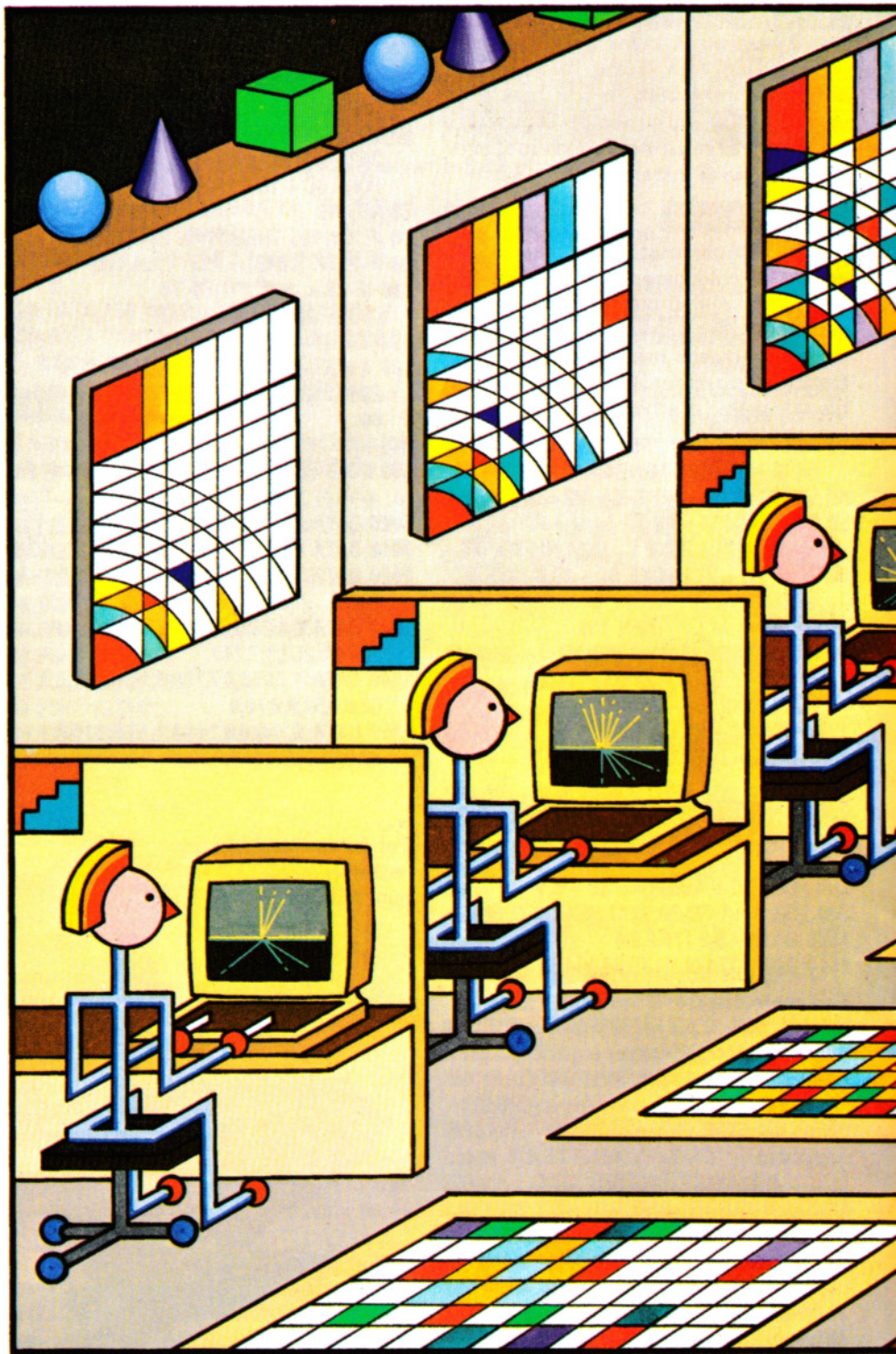
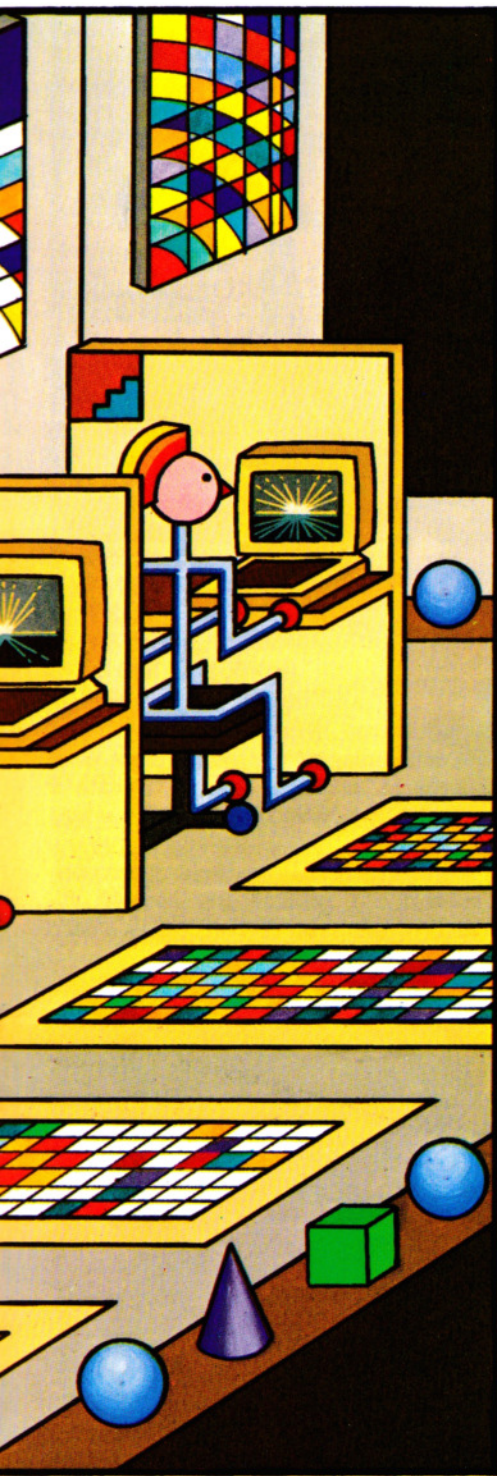
e via dicendo. Il computer, a ogni richiesta così formulata, visualizzerebbe il risultato. Un simile sistema, però, è quantomai faticoso.

Per un uso più efficiente delle capacità del computer si scriva invece:

```
10 FOR n=1 TO 100
20 PRINT n, SQR n
30 NEXT n
40 PRINT "ecco fatto!"
```

```
10 FOR N=1 TO 100
20 PRINT N;"□";SQR(N)
30 NEXT N
40 PRINT "ECCO FATTO!"
```

```
10 FOR N=1 TO 100
20 PRINT N, SQR (N)
30 NEXT N
40 PRINT "ECCO FATTO!"
```

Il programma fa sì che il computer visualizzi, automaticamente, il numero 1 e la sua radice quadrata, poi il numero 2 e la

■ IL COMPUTER CONTA DA 1 a 100
■ I CICLI **FOR ... NEXT** PER CREARE PAUSE
■ LA CREAZIONE DI EFFETTI AUDIO
■ DIPINGERE COI NUMERI
■ UN'AURORA MULTICOLORE COI CICLI **FOR ... NEXT**
■ PATCHWORK E RICAMI COMPUTERIZZATI

sua radice e così via, fino ad arrivare al numero 100.

Come ciò possa accadere è presto detto. Quando il BASIC incontra l'istruzione FOR, sa che le successive linee di programma (comprese tra la FOR e la NEXT più 'vicina') devono essere ripetute un certo numero di volte (nel nostro esempio devono essere ripetute 100 volte).

Cosicché, la linea 20 viene eseguita una prima volta, calcolando la radice quadrata di 1, poi una seconda volta, calcolando la radice quadrata di 2, poi una terza volta calcolando la radice quadrata di 3 e così via di seguito.

Quando il programma raggiunge il massimo valore (specificato dopo la FOR), l'esecuzione prosegue dalla linea 40.

I VALORI FRAZIONARI

Un ciclo FOR ... NEXT può anche procedere per 'passi' diversi da 1.

Si provi ad esempio:

```
10 FOR n=1 TO 30 STEP 2.7
20 PRINT n, SQR n
30 NEXT n
```

```
10 FOR N=1 TO 30 STEP 2.7
20 PRINT N;"□";SQR(N)
30 NEXT N
```

```
10 FOR N=1 TO 30 STEP 2.7
20 PRINT N, SQR(N)
30 NEXT N
```

Si noterà che il computer non si preoccupa del fatto che 30 non sia divisibile per il passo richiesto, ma che, raggiunto il valore più vicino, il programma si arresta.

Un'altra cosa, alla quale il computer non presta attenzione, è il numero di linee comprese tra FOR e NEXT: le esegue tutte, quante esse siano.

AZIONE DI RITARDO

I cicli FOR ... NEXT vengono utilizzati per moltissimi scopi.

Uno di questi è semplicemente quello di... perdere tempo!

Si veda, a questo proposito, il programma delle tabelline a pagina 7.

In quel caso, il ciclo FOR ... NEXT non fa altro che contare ed il risultato più appariscente è una pausa nell'esecuzione. I computer contano molto rapidamente e bastano poche frazioni di secondo (il preciso intervallo di tempo varia da computer a computer) per fare una semplice somma.

Per ottenere una pausa che sia apprezzabile, è necessario fargli contare almeno fino a 1000.

Se però eseguiamo il ciclo riportato di seguito:

```
10 FOR N=1 to 1000000
20 NEXT N
```

con tutta probabilità avremo il tempo di uscire comodamente per berci un buon caffè!

Come fare per tenere il conto delle variabili impiegate in un programma ed evitare doppioni?

Come regola generale, un listato di programma è più chiaro se alle variabili vengono assegnati nomi composti da più lettere, facili da ricordare e di senso compiuto. Ciò è particolarmente vero nei programmi più lunghi. Scrivere FOR riga ... o FOR colonna ... è più facilmente interpretabile di FOR x ... o FOR y. Alcuni BASIC, però, sono in grado di riconoscere soltanto nomi composti da non più di due lettere: in questo caso, può esser utile annotarsi, sopra un foglietto a parte, il nome di ciascuna variabile ed una breve descrizione del suo impiego.

RALLEGRIAMO LE PAUSE

Talvolta, i programmatori, per rendere meno noiose le pause, le 'allietano' con qualche sottofondo musicale. Si provi:

```
10 FOR n=29 TO 10 STEP -1
20 BEEP .015, n
30 NEXT n
```

```
10 FOR N=160 TO 100 STEP -4
20 SOUND 1,-15, N, 1
30 NEXT N
```

```
10 FOR N=12 TO 1 STEP -1
20 PLAY "T40;04;"+STR$ (N)
30 NEXT N
```

Suoni di questo tipo vengono spesso usati per avvertire di un errore, oppure, nei giochi, per segnalare l'atterraggio di una navetta spaziale. Da notare che, quando il conto avviene alla rovescia, il passo STEP deve essere negativo (nel nostro esempio è −1).

(Il Commodore 64 non possiede comandi elementari per i suoni e occorrerebbe scrivere un apposito sottoprogramma, richiamato mediante una GOSUB inserita alla linea 2Ø).

Usiamo il ciclo giusto

L'uso dei cicli FOR ... NEXT è assai frequente, ma altrettanto frequenti sono i casi nei quali è bene evitarne l'uso. La regola generale è:
quando si desidera far ripetere un gruppo di istruzioni un numero prefissato di volte, senza interruzioni di sorta, allora deve essere impiegato un ciclo FOR ... NEXT.

Quando, invece, si desidera far ripetere una certa sequenza fino al verificarsi o meno di una condizione, è preferibile utilizzare strutture del tipo WHILE ... WEND oppure REPEAT ... UNTIL (se il BASIC non consente queste strutture, è necessario ricorrere all'istruzione GOTO, che è posta all'interno del ciclo).

DIPINGERE COI NUMERI

Un po' per gioco e un po' per imparare qualcosa ancora sui cicli, vediamo come ottenere degli effetti grafici utilizzando i FOR ... NEXT. Ecco un esempio:

```
10 FOR n=0 TO 21
20 LET m=RND*31
30 INK RND*7+1
40 PRINT AT n,m; "■"
50 NEXT n
60 GOTO 10
```

```
8 MODE 2
9 VDU23;8202;0;0;0;
10 FOR riga=0 TO 30
20 LET colonna=RND(19)
30 COLOUR 128+RND(7)
40 PRINT TAB(colonna,riga)"□";
50 NEXT riga
60 GOTO 10
```

```
7 CLS0
10 FOR N=0 TO 63
20 LET M=RND(32)-1
30 LET C=RND(9)-1
40 SET (N,M,C)
50 NEXT N
60 GOTO 10
```

In questo caso, la linea 1Ø fissa la profondità dell'immagine che verrà creata sullo schermo e, al tempo stesso, avverte il computer che verrà visualizzata una riga alla volta.

La linea 2Ø fissa invece la larghezza dell'immagine e, assieme alla linea 4Ø, comanda al computer di visualizzare a caso dei quadratini colorati.

La linea 3Ø genera una colorazione casuale dei quadratini.

```
10 PRINT "[CLR]"
15 FOR N=0 TO 24
20 M=INT(RND(1)*18)
25 C=INT(RND(1)*40)
30 POKE 1024+(40*N)+C,160
40 POKE 55296+(40*N)+C,M
50 NEXT
60 GOTO 15
```

Il programma per il Commodore funziona in modo leggermente diverso: la linea 15 'conta' le righe dello schermo, la linea 2Ø sceglie (a caso) la quantità di quadratini da visualizzare su ciascuna riga, mentre

la linea 30 seleziona, sempre a caso, la colorazione dei quadratini.

VARIAZIONE SUL TEMA

Per familiarizzarsi sia con i cicli FOR ... NEXT che con la funzione RND, si può tentare di modificare i programmi appena visti. Ecco, tra le tante possibili, due variazioni (non si dimentichi d'impartire il comando NEW quando s'imposta un nuovo programma):

```
10 FOR n=0 TO21
20 FOR m=0 TO 31
30 INK RND*7+1
40 PRINT AT n,m;"■"
45 NEXT m
50 NEXT n
60 GOTO 10
```

```
10 LET n=RND*21
20 FOR m=0 TO 31
30 INK RND*7+1
40 PRINT AT n,m; "■"
45 NEXT m
60 GOTO 10
```

```
8 MODE 2
9 VDU 23;8202;0;0;0;
10 FOR riga=0 TO 30
20 FOR colonna=0 TO 19
30 COLOUR 128+RND(7)
40 PRINT TAB(colonna,riga)"□";
45 NEXT colonna
50 NEXT riga
60 GOTO 10
```

```
8 MODE 2
9 VDU 23;8202;0;0;0;
10 LET riga=RND(31)-1
20 FOR colonna=0 TO 19
30 COLOUR 128+RND(7)
40 PRINT TAB(colonna,riga)"□"
50 NEXT colonna
60 GOTO 10
```

```
8CLS0
10 FOR N=1 TO 60
20 FOR M=0 TO 31
30 LET C=RND(9) -1
40 SET (N.M.C)
45 NEXT M
50 NEXT N
60 GOTO 10
```

```
8CLS0
10 LET N=RND(60)
20 FOR M=0 TO 31
30 LET C=RND(9) -1
40 SET (N.M.C)
45 NEXT M
60 GOTO 10
```

```
10 PRINT "♡"
20 FOR M=0 TO 999
30 LET C=INT(RND(1)*16)
40 POKE 1024+M,160
50 POKE 55296+M,C
60 NEXT M
70 GOTO 20
```

```
10 PRINT "♡"
20 LET N=INT(RND(1)*25)*40
30 FOR M=N TON+39
40 LET C=INT(RND(1)*16)
50 POKE 1024+M,160
60 POKE 55296+M,C
70 NEXT M
80 GOTO 20
```

Le istruzioni del BASIC, quali ad esempio PRINT, devono essere sempre digitate maiuscole?

In genere sì, tranne nel Commodore, dove si usano le maiuscole se si è in 'modo maiuscolo', le minuscole se si è in 'modo text'.

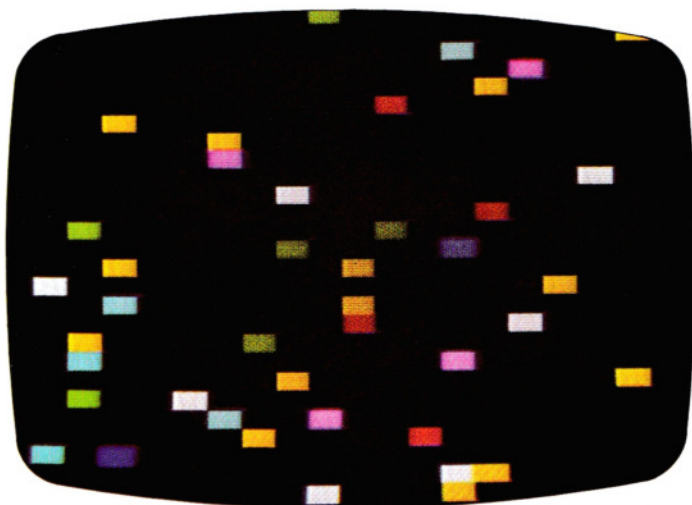
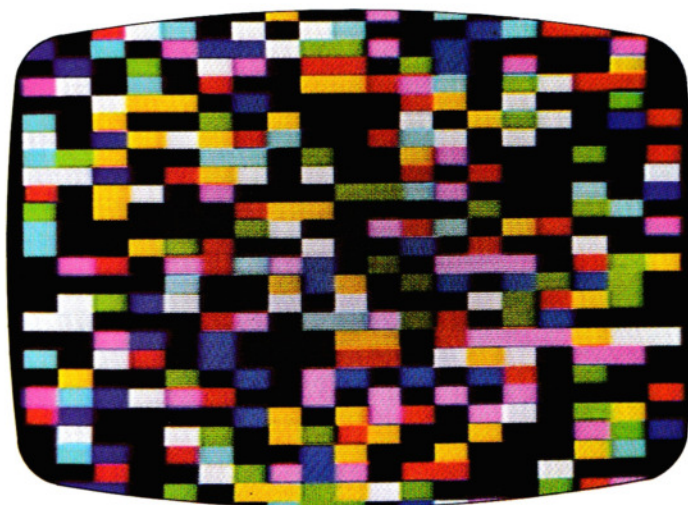
Prima di proseguire, ecco un piccolo esperimento che andrebbe fatto sullo Spectrum, sugli Acorn e sul Dragon: si cancelli la linea 45 dal primo dei due programmi e si esegua di nuovo il programma dopo le seguenti modifiche:

```
55 NEXT m
```

```
55 NEXT colonna
```

```
55 NEXT M
```

Questo serve a scoprire l'ultima importante caratteristica dei cicli FOR ... NEXT: se ne esiste più d'uno, nel medesimo programma, ciascun ciclo deve essere *completamente contenuto* (il termine esatto è *nidificato*) all'interno di un altro. Se due cicli si sovrappongono, il programma non funziona.

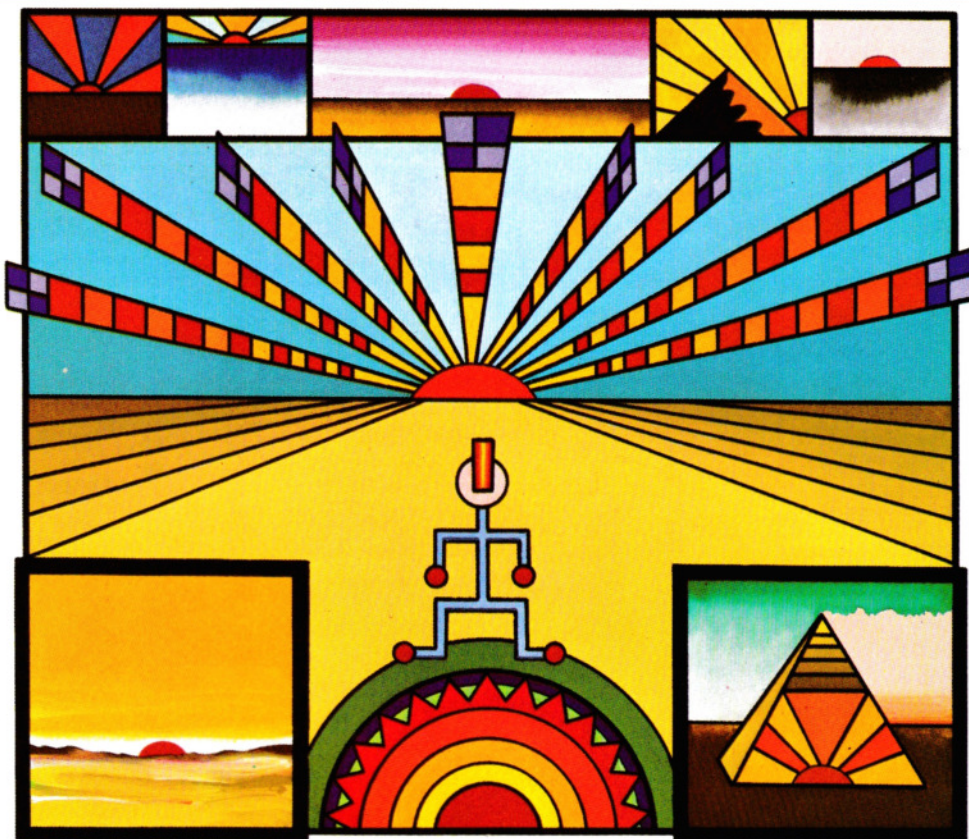
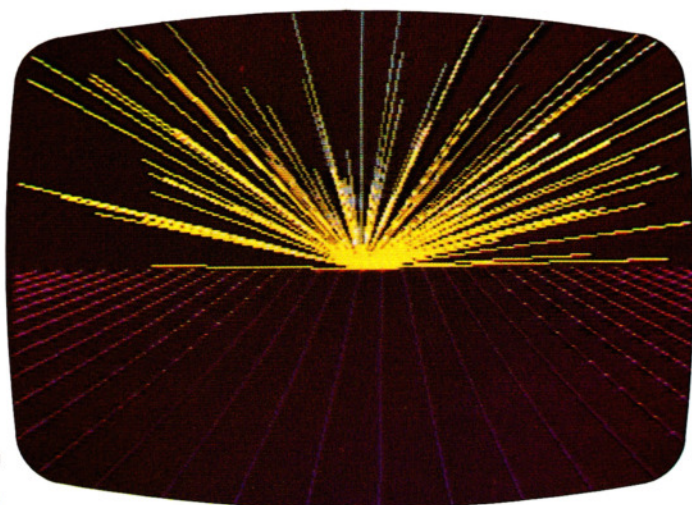
Sul Commodore, nel caso specifico di questo programma, un simile accorgimento è superfluo, ma la regola generale è comunque valida.

1. Prima fase del 'patchwork' sullo schermo...

2. ... che va ripetendosi. (Versione Micro BBC)

UN'AURORA MULTICOLORE

Questo programma utilizza un ciclo FOR ... NEXT per generare l'immagine di un'aurora.

Nella prima parte viene fissata l'origine, sullo schermo, dalla quale partono i 'raggi solari', nella seconda parte del programma, invece, vengono tracciate le linee prospettiche di sottofondo.

```
5 BORDER 0:PAPER 0:INK 6:CLS
10 FOR n=1 TO 80
20 PLOT 127,75
30 DRAW INT (RND*250)—125,INT(RND*97)
40 NEXT n
45 INK 5
50 FOR n=75 TO 0 STEP—15
60 PLOT 127,75
70 DRAW—127,—n: PLOT 127, 75: DRAW
   127,—n
80 NEXT n
100 FOR n=— 127 TO 127 STEP 20
110 PLOT 127,75
120 DRAW n,—75
130 NEXT n
```

```
10 MODE1
15 GCOL0, 2
20 FOR S=1 TO 80
30 LET X=RND(1280)
40 LET Y=512+RND(512)
50 MOVE 640,512
60 DRAW X,Y
70 NEXT S
75 GCOL0, 1
80 FOR L=0 TO 1280 STEP 40
90 MOVE L,512
100 DRAW(L—512)*4,0
110 NEXT L
```

```
20 PMODE 3,1
30 PCLS 3
40 COLOR 2
50 SCREEN 1,0
60 FOR N=1 TO 80
70 LINE(127,95)—(256—RND(256),96
   —RND(96)),PSET
80 NEXT N
90 COLOR 4
100 FOR N=95 TO 191 STEP 12
110 LINE(127,95)—(0,N),PSET
120 LINE(127,95)—(255,N),PSET
130 NEXT N
140 FOR N=0 TO 255 STEP 10
150 LINE (127,95)—(N,191), PSET
160 NEXT N
170 GOTO 170
```

3. Il levarsi del sole ... secondo gli Acorn

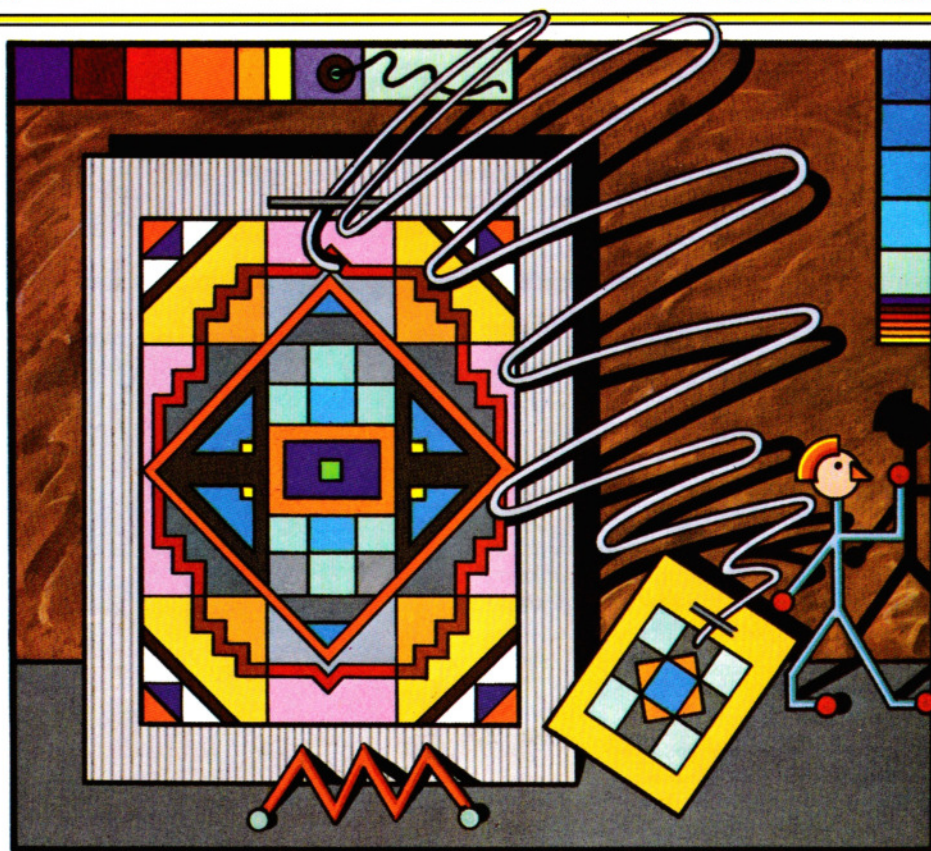
4. Un bel ricamo eseguito al ... BBC!

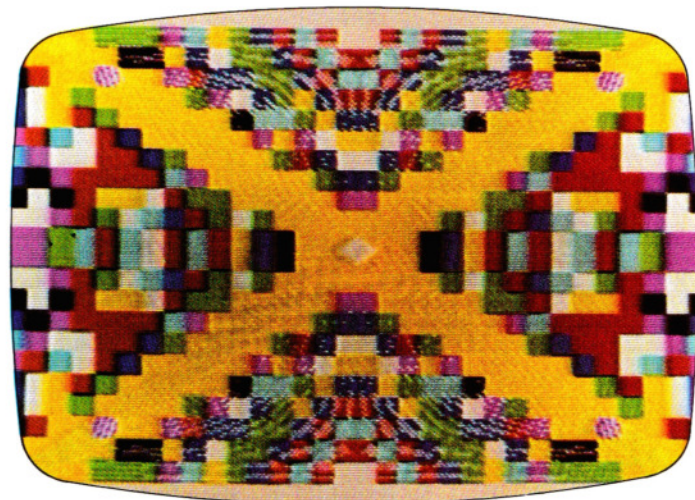
RICAMI... ISTANTANEI

Infine, ecco un programma veramente spettacolare per tre dei nostri computer:

```
10 FOR n=0 TO 255 STEP 2
15 INK RND*8
20 PLOT 0,0: DRAW n,175
30 PLOT 255,0: DRAW −n,175
40 PLOT 0,175: DRAW n, −175
50 PLOT 255,175: DRAW −n,−175
60 NEXT n
70 GOTO 10
```

```
8 MODE 2
10 FOR N=0 TO 1279 STEP 10
15 GCOL 0,RND(7)
20 MOVE 0,0: DRAW N,1023
30 MOVE 1279,0: DRAW 1279−N,1023
40 MOVE 0,1023: DRAW N,0
50 MOVE 1279,1023: DRAW 1279−N,0
60 NEXT N
70 GOTO 10
```

```
3 PMODE 3,1
6 PCLS
9 SCREEN1,0
10 FOR L=0 TO 255 STEP 2
15 COLOR RND(4)
20 LINE (0,0)−(L, 191),PSET
30 LINE (255,0)−(255−L, 191),PSET
40 LINE(0,191)−(L,0),PSET
50 LINE(255,191),−(255−L,0),PSET
60 NEXT L
70 GOTO 10
```

Ciascuno dei quattro segmenti che compongono l'immagine inizia con un sol punto, posto in un angolo dello schermo. Il ciclo FOR ... NEXT, in questo caso, serve per costruire le linee colorate che attraversano lo schermo. Come funzioni esattamente l'impostazione grafica di questo programma sarà chiaro più avanti nel corso, ma si provi a eliminare le linee 30 e 40 ...

Altri esperimenti possono essere i seguenti: eliminare il tracciamento di alcune delle righe, variare i colori impiegati o il valore dello STEP nella linea 10, omettere la GOTO alla fine del programma. In pochi minuti si possono ottenere centinaia di 'motivi' originali degni delle più note case di moda! Questi esperimenti, oltre a esser più o meno divertenti, insegnano anche ad abituarci all'uso delle varie istruzioni grafiche.

5. La versione su Spectrum è più stilizzata...

6. ... ma le variazioni sono infinite!

COME USARE LE SAVE E LE LOAD

Non c'è niente di più irritante di un programma che non si riesce a memorizzare o di un gioco che non si riesce a leggere dal nastro. Consigli per ridurre al minimo la frustrazione

Poche altre funzioni hanno un impatto così immediato sul principiante, quanto quelle relative alla memorizzazione e alla rilettura dei programmi sul nastro magnetico del registratore.

Tutti, le prime volte, incontrano qualche problema, le cui cause non sono sempre facilmente individuabili, anche avendo qualche dimestichezza coi computer. Né, del resto, acquisita una certa pratica, si eliminano del tutto gli inconvenienti dovuti a registrazioni non perfette. Tuttavia, è possibile minimizzare i rischi adottando metodi sistematici.

LE CORRETTE CONNESSIONI

In alcuni computer il registratore è incorporato e, almeno in questo caso, non dovrebbero sorgere problemi di connessioni. Ma, nella maggior parte dei casi, il registratore va collegato al computer con uno o più cavetti. In genere, gli spinotti applicati ai cavetti sono: uno del tipo DIN (dal lato del computer) e tre piccoli *jack* (dal lato del registratore). Ma non sempre è così.

Ovviamente, nell'acquistare o nel fabbricarsi i cavetti di raccordo, occorre assicurarsi di adoperare gli spinotti giusti!

IL CONTROLLO DEI LIVELLI

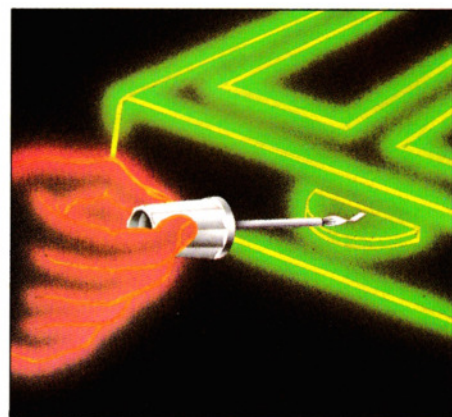
Una delle prime operazioni, tra le più importanti per il successo di future SAVE e LOAD, è quella di stabilire con esattezza la posizione dei controlli di tono e di volume sul registratore impiegato. I manuali di alcuni computer sono abbastanza generosi nel fornire consigli in merito, nel qual caso è bene attenersi scrupolosamente a quanto indicato. Altri, invece, trattano in modo sommario l'argomento, lasciando nei guai il povero utente.

Se si adopera un normale registatore domestico, è buona norma escludere tutte le varie funzioni speciali (attenuatori, filtri ecc.) e posizionare il controllo dei toni sugli 'alti' ogni volta che lo si usa col computer.

La regolazione del controllo di volume va, purtroppo, trovata empiricamente: si porti il regolatore di volume a circa metà percorso e si provi a leggere un programma da un nastro preregistrato (la cassetta dimostrativa venduta con il computer, ad esempio). Se il tentativo fallisce, riprovare aumentando leggermente, di volta in volta, il livello del volume.

Se, raggiunto il livello massimo, non si è ancora ottenuto alcun successo, si riparta dalla posizione centrale, ma questa volta diminuendo di passo in passo il volume.

Se anche l'esito di questa prova è negativo, si ricontrollino i collegamenti, o si provi a sostituirli con altri, o, addirittura, si ricorra a un altro nastro o a un altro registratore. Come ultima risorsa, non rimane che rivolgersi direttamente al venditore del computer.

Se, viceversa, si è trovato il corretto li-

1. Se possibile, marcare la posizione ottimale del livello di volume

- LE CORRETTE CONNESSIONI
- IL CONTROLLO DEI LIVELLI
- IL COMANDO **SAVE**
- IL PROCESSO DI VERIFICA
- IL COMANDO **LOAD**

vello di volume, è bene annotarselo (o marcarlo in qualche modo). Ci sarà utile, in seguito, per non ripetere nuovamente tutto questo lungo procedimento. Generalmente, per quando concerne la registrazione, quasi tutti gli attuali apparecchi sono dotati di un livellamento automatico, permettendo così di lasciare inalterato il controllo di volume.

Per una sommaria verifica, si azzeri la memoria del computer (impartendo un NEW, seguito da ENTER o RETURN), si trascriva il seguente programma:

```
10 REM TEST
20 REM
30 REM
40 REM
```

e si digiti il comando SAVE, come specificato nel manuale dell'apparecchio. Una volta memorizzato il programma ed impartito un NEW, si provi l'istruzione LOAD: se tutto è a posto, il programma può adesso essere esaminato con un LIST.

IL COMANDO SAVE

Il formato del comando SAVE varia a seconda della marca del computer, ma vi sono delle operazioni preliminari di carattere generale. Prima di usare il comando SAVE, per memorizzare un programma, assicurarsi che nel registratore sia caricata una cassetta adatta all'uso. Riavvolgere opportunatamente il nastro, facendo attenzione a posizionarlo dopo la 'coda' iniziale, non magnetica.

Quindi, viene la volta di scegliere un nome per il *file* (archivio) che intendiamo memorizzare. Tutte le informazioni vengono registrate sotto forma di file, siano esse istruzioni di un programma o dati. La scelta del file, da impiegarsi assieme al comando SAVE, deve seguire alcune semplici regole. In genere, sono ammessi nomi composti da una decina di caratteri alfanumerici (il primo deve essere, però, alfabetico).

Abbiamo ampia libertà di scelta, quindi, purché non superiamo il numero massimo di caratteri e racchiudiamo il nome tra due doppi apici, ad esempio:

```
SAVE "NOMEPROG01"
SAVE "NomeProg01"
```

Ambedue sono nomi adatti. Nel secondo esempio, però, viene fatto un uso misto di maiuscole e minuscole e questo può rivelarsi un'arma a doppio taglio. Infatti, quando in seguito rileggeremo il file, questo nome dovrà esser riscritto con assoluta esattezza nel formato minuscolo/maiuscolo!
Altri esempi di nomi idonei sono:

```
SAVE "NOME.PROG"
SAVE "NOME/PROG"
SAVE "NOME(PROG)"
```

In questi vengono adoperati caratteri non

TROUBLE SHOOTER

- Si usi un semplice e affidabile registratore portatile monofonico, possibilmente riservandolo esclusivamente all'impiego col computer. Per mantere costante la velocità, usare la tensione di rete
- Si eviti l'uso di complessi stereo hi-fi, dove, paradossalmente, i troppi comandi possono non garantire un corretto funzionamento dei trasferimenti.
- Si usino soltanto nastri di buona qualità: trovata una buona marca, si eviti di cambiarla per altre. I programmi utili vengono usati spesso e nastri di bassa qualità si deteriorano rapidamente
- Si osservino rigorosamente le istruzioni (spesso riportate anche sullo schermo) durante la lettura e la scrittura
- Una volta individuati i livelli di volume e di toni ottimali, si annotino le rispettive posizioni dei comandi
- Si allontani il registratore da probabili fonti di rumore (apparecchi TV, monitor del computer, ecc.) qualora non si riuscisse a leggere un programma precedentemente usato con successo. Si provi, eventualmente, a caricare un altro programma. Se questo viene letto regolarmente, il primo nastro potrebbe essere danneggiato.
- Se si trova necessario variare il livello di volume, da programma a programma, conviene annotarsi il livello richiesto sulla cassetta
- Si prenda l'abitudine di ravvolgere i nastri dopo l'uso. Spesso capita di avviare una lettura su una parte vuota del nastro. Si inizi la ricerca dei programmi dal principio del nastro (i contagiri non sono mai molto precisi)
- Conservare i nastri in un luogo asciutto, lontano da fonti magnetiche o di calore e dalla polvere
- Attenzione a non eseguire un LOAD avendo ancora un programma da salvare in memoria

Importanza della pulizia nei registratori usati con i computer

Una regolare manutenzione del registratore, che richiede solo qualche minuto, è forse più importante che per gli apparecchi usati per riprodurre musica.

Una delle più frequenti cause d'insuccesso, nel trasferimento di dati, sta proprio in questo genere di trascuratezze.
Si usi una delle tante buone confezioni per la manutenzione dei registratori a cassette, onde rimuovere gli immancabili depositi di ossido che si formano sulle testine magnetiche durante l'uso.
Attenzione a non graffiare le testine con oggetti appuntiti.
Si usino solo, con estrema cura, gli appositi solventi (le parti in gomma potrebbero restare danneggiate).

alfabetici per separare due parti del nome.

Se il computer impiegato è fornito di controllo a distanza, si può predisporre l'apparecchio per la registrazione (tasti RECORD + PLAY, in genere). A questo punto, digitato il comando SAVE seguito dal nome scelto e da un ENTER o RETURN, il computer provvede automaticamente al funzionamento del motore del registratore, finché non termina il 'salvataggio' del programma.

Se non esiste questo tipo di automatismo, allora si avvia manualmente il registratore e si impartisce il comando SAVE. Solitamente, il computer, per informare che la memorizzazzione è giunta al termine, visualizza un qualche simbolo sullo schermo.

Se il programma (o la serie di dati) è di particolare importanza, non è male farne una seconda copia, su di un'altra cassetta, ripetendo la medesima procedura.

Il tempo occorrente per il caricamento di un programma dipende da due fattori: le sue dimensioni, in termini di occupazione di memoria e la velocità di trasferimento dei dati tra computer e registrare (questa è variabile su alcuni apparecchi, fissa su altri).

In genere, le migliori registrazioni si hanno a velocità di trasferimento basse; con alte velocità si usa meno nastro e l'operazione richiede minor tempo, ma in tal caso occorre usare un nastro di buona qualità e un buon registratore.

Alcune varianti al comando SAVE verranno spiegate più avanti nel corso.

Annotare immediatamente, sulla stessa cassetta, il nome del file salvato è un'abitudine molto consigliata, ma troppo spesso poco seguita, cosicché si finisce col dimenticare su quale nastro si trovi un programma, se non, addirittura il suo nome!

Volendo, i programmi su cassetta si possono proteggere contro involontarie cancellature, semplicemente rimuovendo l'apposita linguetta di plastica.

IL PROCESSO DI VERIFICA

Un metodo molto sbrigativo per verificare la corretta memorizzazione su nastro di un programma è di rileggerlo con un LOAD e di tentare l'esecuzione. Ma questo è un metodo davvero sconsigliabile, poiché il comando LOAD cancella il programma preesistente in memoria, ossia l'originale. Qualora la registrazione fosse andata male ... dovremmo riscrivere tutto il programma!

Molti computer sono dotati di uno speciale comando VERIFY, che, come si può in-

CAVI SCHERMATI A 6 POLI

PRESA DIN A 7 POLI

PRESA DIN A 7 POLI

2. Ecco due tipi molto comuni di cavi di collegamento tra computer e registratore. Dovendo costruirseli da sé, si controlli l'esatta disposizione delle connessioni, specialmente del lato del computer (a sinistra)

PRESA JACK mm 2,5 (CONTROLLO MOTORE)
PRESA JACK mm 3,5 (INGRESSO REGISTRATORE)
PRESA JACK mm 3,5 (USCITA REGISTRATORE)
PRESA JACK mm 2,5 (CONTROLLO MOTORE)
MAGLIA
PRESA DIN A 5 POLI

tuire, serve a verificare l'esattezza della copia su nastro, confrontandola con quella presente in memoria, dopo che si è fatto un SAVE. Prima di impartire il comando VERIFY, occorre riavvolgere il nastro e usare VERIFY assieme al nome del file salvato, ad esempio:

```
VERIFY "NOMEPROG01"
```

Il computer informa, a tempo debito e con opportuni messaggi, sull'esito dell'operazione. In caso di errore, si ha così la possibilità di ripetere, eventualmente, la memorizzazione fino ad essere certi del risultato.

IL COMANDO LOAD

Se si comincia usando cassette preregistrate, i primi problemi si incontrano con il comando LOAD, che serve per leggere dai nastri. Anche il comando LOAD può avere molteplici formati, ma per adesso vediamo il più semplice:

```
LOAD "NOMEPROG01"
```

che è l'esatto corrispondente di quello usato per SAVE e VERIFY. Ovviamente, se il nome del file è sbagliato, la lettura non avrà alcun successo.

Generalmente, dopo aver impartito il comando LOAD, si avvia il nastro. Se vi sono più programmi, il computer ignora tutti quelli che incontra, finché non trova il file col nome specificato, caricandolo in memoria.

Se il registratore è controllato a distanza dal computer, il motore si arresta automaticamente a lettura avvenuta; in tutti i casi, comunque, occorre, a questo punto, premere il tasto STOP del registratore.

Il programma caricato con il comando LOAD sostituisce quello presente in memoria (qualora ve ne fosse uno), pertanto occorre prestare attenzione a non cancellare irremidiabilmente qualcosa d'importante senza prima averlo memorizzato.

Adesso si provi a riavvolgere il nastro sul quale si è registrato il programmino di prova usato per il comando SAVE e a sperimentare un LOAD. In caso di problemi, si consulti la rubrica FONTI D'ERRORE.

Altri formati del comando LOAD vengono usati per accedere a programmi in codice macchina, per spostare l'origine di un programma eseguibile e per fondere assieme più programmi. Alcune di queste varianti s'incontrano spesso nelle istruzioni di caricamento di programmi per giochi. Esamineremo più avanti alcune di queste varianti nonché il loro impiego più frequente.

Si possono usare vecchi nastri musicali per registrare i dati?

Benché sia preferibile impiegare nastri appositamente studiati per computer, nulla vieta di usare nastri musicali usati di buona qualità, purché siano stati trattati con la dovuta cura.
Qualche lieve imperfezione del nastro, anche se appena percettibile in una riproduzione musicale, può, tuttavia, compromettere irrimediabilmente l'esattezza dei dati o il funzionamento di un intero programma. Basta, infatti, che un singolo dato sia danneggiato per impedire la lettura di un programma. Un'altra possibile fonte di inconvenienti è che spesso i segnali precedentemente registrati tendono ad affiorare, essendo stati cancellati solo parzialmente durante la registrazione dei dati.

QUALCHE MOSSA D'ANIMAZIONE

Vogliamo aggiungere un po' di vita ai nostri programmi di giochi? Tanto per cominciare, ecco alcuni caratteri grafici di facile impiego

Usare i programmi di giochi commerciali è senza dubbio divertente, ma viene sempre il momento in cui si desidera dare briglia sciolta alla propria fantasia nella creazione di nuovi giochi.

La programmazione dei giochi non è facile: occorre partire dagli aspetti più elementari, anche se si è impazienti di ottenere subito effetti grandiosi. Così facendo, si imparerà a pensare in termini logici rafforzando la propria capacità di programmare.

La prima cosa da imparare, oltre ovviamente al linguaggio BASIC, è la tecnica d'animazione.

Per creare l'illusione del movimento, il programmatore di giochi usa tecniche molto simili a quelle impiegate dai disegnatori di cartoni animati. Il ritmo ideale, nel susseguirsi di immagini leggermente diverse l'una dall'altra, è di 24 volte al secondo.

Nei film, però, c'è un vantaggio: quando compare l'immagine successiva, quella precedente si 'cancella' automaticamente dallo schermo, mentre non è così sul computer.

Uno dei sistemi più semplici, se ci si vuole liberare di un'immagine inutile, è quella di visualizzare, al suo posto, un altro carattere.

Ad esempio, se la linea 10 di un programma provoca la visualizzazione della lettera A in una certa posizione e, dopo qualche altra istruzione, viene visualizzata nella medesima posizione la lettera B, la A scompare.

Nei programmmi che seguono è stata adottata in larga misura questa tecnica 'sostitutiva'.

Volendo semplicemente cancellare la A, possiamo visualizzare al suo posto uno spazio, "□".

Dimenticando un simile accorgimento, lo schermo finirà ben presto di riempirsi di caratteri indesiderati, rovinando la nostra fatica.

Esistono differenze molto grandi tra i sistemi grafici di computer appartenenti a marche diverse.

Anche in quelli che usano caratteri predefiniti (memorizzati nelle ROM) le differenze sono notevoli.

Un semplicissimo esempio di animazione, sul Dragon, si ottiene col seguente programma, nel quale è simulato un 'millepiedi' (vedi figura) in movimento:

```
 ) ) )       ) ) )       ) ) )       ) ) )
 OOO <       OOO <       OOO <       OOO <
 ) ) )       ) ) )       ) ) )       ) ) )
```

```
10 PRINT @ 238, "000"
20 PRINT @ 206, ")))"
30 PRINT @ 241, "<"
40 PRINT @ 270, ")))"
```

Queste linee creano il millepiedi: quanto al fatto di muoverlo, ciò richiede un certo impegno, ma è utile a dimostrare alcuni punti interessanti.

In primo luogo, si acquista un po' di pratica sulla visualizzazione di oggetti sullo schermo: l'animaletto è centrato sullo schermo (posizione 239).

In secondo luogo, è illustrato cosa avviene tentando di visualizzare (con la PRINT) più caratteri nella medesima locazione: il computer deposita il carattere nella locazione immediatamente successiva. È questa la ragione per cui i "piedi", alla linea 30, si trovano alla locazione 241: le locazioni da 238 a 240 sono già occupate dal 'corpo'!

Un modo conveniente di definire l'insetto è di concentrare il precedente programma in un'unica linea. Detto per inciso, l'istruzione PRINT, sul Dragon, si può immettere usando un semplice punto interrogativo (?). Quando si lista il programma, compare regolarmente l'istruzione PRINT.

Il programma semplificato diventa:

```
10 PRINT @ 128, "000<":PRINT @ 206,
   ")))";PRINT @ 270, ")))"
```

Con solo altre due linee si ottiene già un po' d'animazione:

```
20 PRINT @ 238, "000<":PRINT @
   206, "(((":PRINT @ 270, "((("
30 GOTO 10
```

Ma eseguendo il programma con un RUN, ci si accorge che il movimento risulta troppo rapido. Possiamo allora inserire dei cicli FOR ... NEXT allo scopo di creare dei ritardi.

Qui usiamo il valore 15 ma, variando questo valore, possiamo variare il 'passo' del millepiedi:

```
15 FOR L=1 TO 15
17 NEXT L
25 FOR L=1 TO 15
27 NEXT L
```

Abbiamo così ottenuto un primo abbozzo d'animazione, che verrà in seguito sviluppato (vedere al paragrafo IL MOVIMENTO).

```
 ) ) )       ) ) )       ) ) )       ) ) )
 OOO <       OOO <       OOO <       OOO <
 ) ) )       ) ) )       ) ) )       ) ) )
```

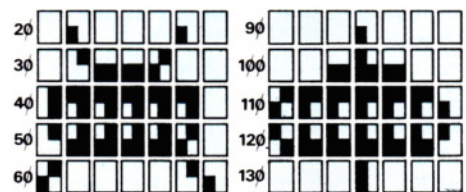
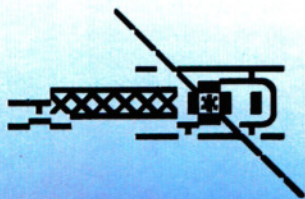
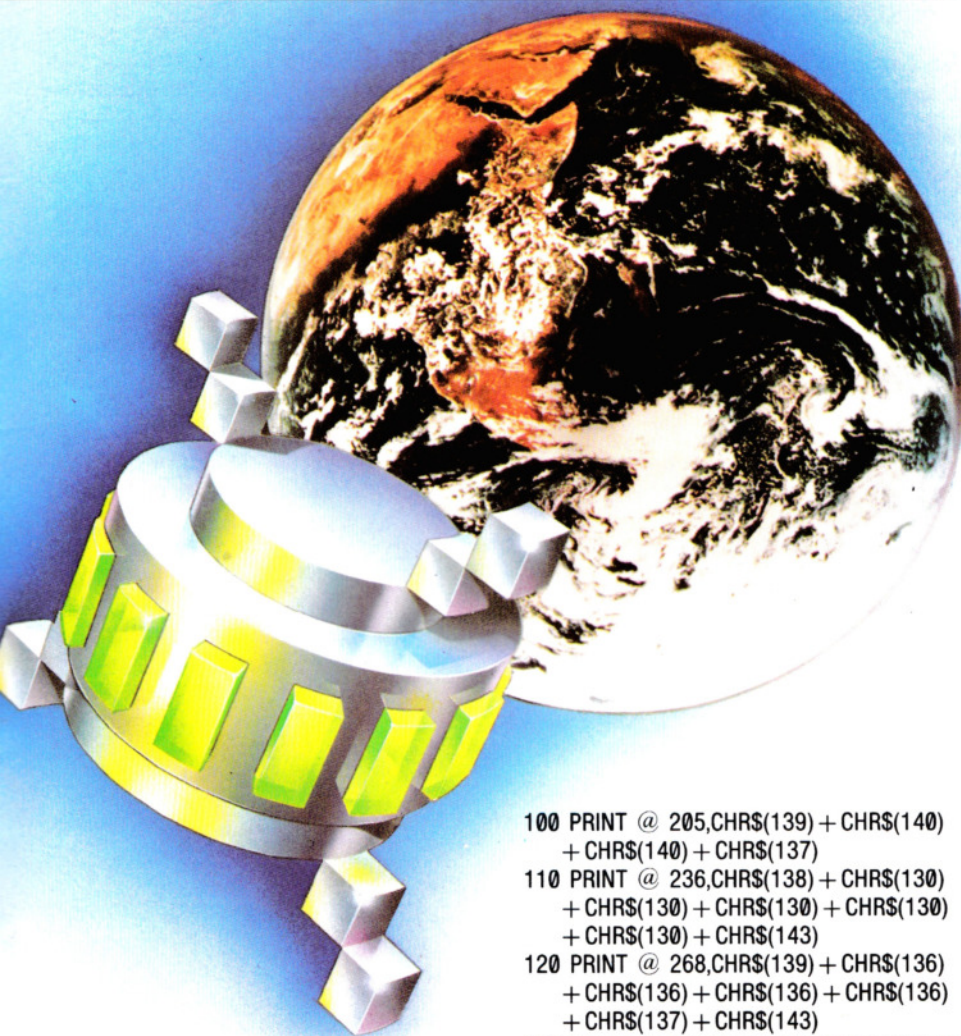
GRAFICA IN BASSA RISOLUZONE

Per creare qualche effetto più interessante, si può ricorrere alla grafica in bassa risoluzione del Dragon. Il manuale di questo apparecchio riporta in una tabella i caratteri grafici a disposizione, ognuno dei quali ha un numero di codice (da 128 a 143). Per visualizzarli, si usa la funzione CHR$, seguita dal codice racchiuso fra parentesi.

Ad esempio, per visualizzare il carattere con codice 138 al centro dello schermo, si scrive:

```
10 PRINT @ 239, CHR$ (138)
```

Il seguente programma fa ruotare l'immagine di un piccolo satellite (figura 1).

```
10 CLEAR 500:CLS
20 PRINT @ 174,CHR$(143)+CHR$(141):
   PRINT @ 178,CHR$(143)
```

1. Come costruire un satellite.

- I PRINCIPI DELL'ANIMAZIONE
- COMANDIAMO I MOVIMENTI CON LA TASTIERA
- COME USARE LA GRAFICA IN ROM

```
30 PRINT @ 206,CHR$(14Ø) + CHR$(132) +
   CHR$(141)
40 PRINT @ 236,CHR$(137) + CHR$(129) +
   CHR$(129) + CHR$(129) + CHR$(129) +
   CHR$(129) + CHR$(141)
50 PRINT @ 268,CHR$(135) + CHR$(132) +
   CHR$(132) + CHR$(132) + CHR$(132) +
   CHR$(133) + CHR$(135)
60 PRINT @ 3ØØ,CHR$(143):PRINT @ 3Ø3,
   CHR$(133):PRINT @ 3Ø5,CHR$(143) +
   CHR$(143)
70 FOR X=1 TO 1Ø
80 NEXT X
90 PRINT @ 173,CHR$(141) @ 177,CHR$
   (141)
100 PRINT @ 2Ø5,CHR$(139) + CHR$(14Ø)
    + CHR$(14Ø) + CHR$(137)
110 PRINT @ 236,CHR$(138) + CHR$(13Ø)
    + CHR$(13Ø) + CHR$(13Ø) + CHR$(13Ø)
    + CHR$(13Ø) + CHR$(143)
120 PRINT @ 268,CHR$(139) + CHR$(136)
    + CHR$(136) + CHR$(136) + CHR$(136)
    + CHR$(137) + CHR$(143)
130 PRINT @ 3ØØ,CHR$(137):PRINT @ 3Ø3,
    CHR$(143):PRINT @ 3Ø5,CHR$(139) +
    CHR$(141)
140 FOR X=1 TO 1Ø
150 NEXT X
160 GOTO 1Ø
```

La linea 1Ø non deve preoccupare. CLEAR 5ØØ serve soltanto per accantonare sufficiente memoria per i caratteri, mentre CLS provoca la pulizia dello schermo.

Si confrontino i caratteri corrispondenti ai vari codici riportati nel programma e si tenti di capire come sia stata costruita l'immagine di figura 1. L'istruzione '+ CHR$()' serve per visualizzare il carattere nella successiva posizione sullo schermo.

IL MOVIMENTO

Ecco, finalmente, un programma che non solo 'anima' l'insetto, ma lo fa muovere sullo schermo:

```
10 CLS
20 FOR N=Ø TO 28
30 PRINT @ 192+N, ")))":
   PRINT @ 224+N,"OOO<":
   PRINT @ 256+N,")))"
40 FOR X=1 TO 1Ø
50 NEXT X
60 PRINT @ 192+N,"□□□":PRINT @ 224
   +N,"□□□□": PRINT @ 256+N,
   "□□□"
70 PRINT @ 192+N,"(((":PRINT @ 224+
   N,"OOO<":PRINT @ 256+N,"((("
80 FOR X=1 TO 1Ø
90 NEXT X
100 PRINT @ 192+N,"□□□":PRINT @
    224+N,"□□□□":PRINT @ 256+N,
    "□□□"
110 NEXT N
120 GOTO 2Ø
```

Nel programma ci sono tre cicli FOR ... NEXT. I due che impiegano la X servono unicamente, come nel primo programma, per creare ritardi. Il ciclo FOR ... NEXT nel quale è usata N, ha invece una funzione diversa, quella di permettere all'animaletto di spostarsi sullo schermo. (Per sapere come funziona un ciclo FOR ... NEXT vedere PROGRAMMAZIONE BASIC 2).

Può meravigliare che la linea 2Ø contenga FOR N=Ø TO 28, quando vi sono 32 posizioni in ciascuna linea dello schermo (numerato da Ø a 31). La ragione di ciò risiede nel fatto che l'insetto è lungo 4 caratteri. Usando un valore maggiore di 28 l'immagine verrebbe spezzettata, tra la fine di una riga e l'inizio di quella successiva, quando raggiunge l'estremità dello schermo.
Ciò dipende dal sistema adoperato per numerare le posizioni sullo schermo: la posizione 32, per esempio, è la prima della seconda riga dall'alto.

Le linee 6Ø e 1ØØ, a prima vista, non sembrano visualizzare un bel niente: in realtà si tratta di una cancellazione secondo la tecnica 'sostituiva' precedentemente descritta.

Si possono creare insetti e mostri, sui computer Acorn, anche usando i semplici caratteri della tastiera, quali le parentesi, i punti e le lettere.

Qui sotto è riprodotto un piccolo millepiedi, facile da 'animare'.

Per prima cosa, lo definiamo in posizione statica:

```
) ) )        ) ) )        ) ) )        ) ) )
OOO <        OOO <        OOO <        OOO <
) ) )        ) ) )        ) ) )        ) ) )
```

```
5 CLS
10 PRINT TAB(15,10);"OOO"
20 PRINT TAB(15,9);")))"
30 PRINT TAB(15,11);")))"
40 PRINT TAB(18,10);"<"
```

(Si noti che in questo programma, dopo la TAB, non ci sono spazi).

Adesso si esegua con un RUN.

Si tratta di un'immagine molto semplice, non tanto somigliante, ma serve a illustrare alcuni punti importanti.

In primo luogo, serve a dare un'idea delle varie posizioni relative sullo schermo. L'insetto si trova in posizione più o meno centrata sullo schermo.

In secondo luogo, illustra l'effetto ottenuto, chiedendo al computer di visualizzare più di un carattere nella medesima posizione sullo schermo: i caratteri vengono semplicemente riprodotti nelle successive locazioni. Ecco perché le 'zampe' (alla linea 4Ø) sono in posizione 18,1Ø: infatti, le locazioni 15,1Ø; 16,1Ø; e 17,1Ø; sono già occupate dal 'corpo' dell'insetto.

Per animare l'immagine, dobbiamo creare un'immagine appena diversa dalla precedente e sovrapporre rapidamente e alternativamente le due.

Aggiungiamo le seguenti linee:

```
50 PRINT TAB(15,10);"OOO"
60 PRINT TAB(15,9);")))"
70 PRINT TAB(15,11);"((("
80 PRINT TAB(18,10);"<"
```

Con l'apporto della seguente linea, si ottiene l'animazione.

```
90 GOTO 10
```

Nell'eseguire il programma, si noterà come il cursore lampeggiante rovini tutto l'effetto. Per ovviare, si aggiunga:

```
7 VDU 23;8202;0;0;0;
```

Se vogliamo cambiare la velocità delle zampette, possiamo usare l'istruzione INKEY, che produce, appunto, un ritardo. Questo è misurato in centesimi di secondo, per cui, scrivendo A = INKEY (1ØØ), otteniamo un ritardo di circa un secondo. Si aggiungano le seguenti linee:

```
45 LET A=INKEY(50)
85 LET A=INKEY(50)
```

Adesso possiamo variare il valore delle INKEY, fino ad ottenere un movimento realistico.

USO DELLA GRAFICA TELETEXT

Se si possiede un Micro BBC, si possono creare immagini migliori con i caratteri grafici (la procedura per l'Acorn Electron verrà trattata in seguito).

Nel MODO 7, il computer BBC è capace di generare tutta una serie di caratteri grafici a blocchi, ottenendo interessanti risultati.

Prima si disegni una sagoma sopra un foglio di carta millimetrata. Poi la si suddivida in rettangoli larghi due quadrati e alti 3.

Ciascun rettangolo si può ricondurre a uno dei caratteri grafici riportati sul manuale del BBC, assieme ai relativi codici.

Se si estraggono i codici, riscrivendoli accanto alla sagoma come ad esempio in figura 2, possiamo disegnare la figura sullo schermo. Ecco un esempio:

```
10 MODE 7
20 LET Y=10; LET X=15
40 VDU 31,X,Y,146,160,160,191,160,160
50 VDU 31,X,Y,+1,146,184,163,255,240,160
60 VDU 31,X,Y+2,146,160,168,189,236,160
70 VDU 31,X,Y+3,146,160,224,165,234,176
80 VDU 31,X,Y+4,146,168,177,160,160,160
90 VDU 31, X,Y,146,160,252,160,160
100 VDU 31,X,Y+1,146,232,236,189,174,
   160
110 VDU 31,X,Y+2,146,162,238,177,160,
   160
120 VDU 31,X,Y+3,146,240,250,162,180,
   160
```

16Ø	16Ø	191	16Ø	16Ø
184	163	255	24Ø	16Ø
16Ø	168	189	236	16Ø
16Ø	224	165	234	176
168	177	16Ø	16Ø	16Ø

16Ø	16Ø	252	16Ø	16Ø
232	236	189	174	16Ø
162	238	177	16Ø	16Ø
24Ø	25Ø	162	180	16Ø
161	16Ø	16Ø	245	16Ø

2. Due immagini di uomo in corsa.

```
130 VDU 31,X,Y+4,146,161,16Ø,16Ø,245,
    16Ø
140 GOTO 4Ø
```

La prima linea selezione il Modo grafico 7 (Teletext). La linea 2Ø stabilisce la posizione dell'omino sullo schermo.

Le linee da 4Ø ad 8Ø definiscono la prima immagine dell'uomo in movimento, mentre le linee da 9Ø a 13Ø lo definiscono nel secondo atteggiamento. Di nuovo, può tornar comodo usare INKEY dopo le linee 8Ø e 13Ø.

Il comando VDU controlla le caratteristiche dello schermo, cosicché VDU 31,X,Y equivale a PRINT TAB (X,Y). Il successivo valore seleziona il colore: nel nostro caso l'omino è colorato in verde (valore 146). I valori successivi sono quelli che controllano la forma.

IL MOVIMENTO

Far muovere l'immagine sullo schermo è abbastanza semplice. Basta iniziare da una posizione X=Ø, alla sinistra sullo schermo, e muoverla, di una posizione alla volta, verso destra. Col computer, ciò si ottiene uando un ciclo FOR ... NEXT. Occorre aggiungere, quindi:

```
3Ø FOR X=Ø TO 35
14Ø NEXT X
```

Ciò fa muovere l'omino dalla colonna Ø alla 35.

Sebbene vi siano 4Ø colonne sullo schermo (da Ø a 39), limitiamo il ciclo a 35, in quanto la larghezza dell'immagine è appunto di cinque colonne. Si provi, a titolo d'esperimento, a modificare la linea 3Ø.

Ora introduciamo un leggero ritardo:

```
85 LET A=INKEY(3)
135 LET A=INKEY(3)
```

Infine, aggiungiamo ancora una linea, per eliminare il cursore lampeggiante:

```
15 VDU 23;82Ø2;Ø;Ø;Ø;
```

Ecco, quindi, il programma completo:

```
1Ø MODE7
15 VDU 23;82Ø2;Ø;Ø;Ø;
2Ø LET Y=1Ø
3Ø FOR X=1 TO 35
4Ø VDU 31,X,Y,146,16Ø,16Ø,191,16Ø,16Ø
5Ø VDU 31,X,Y+1,146,184,163,255,24Ø,16Ø
6Ø VDU 31,X,Y+2,146,16Ø,168,189,236,16Ø
7Ø VDU 31,X,Y+3,146,16Ø,224,165,234,176
8Ø VDU 31,X,Y+4,146,168,177,16Ø,16Ø,16Ø
85 LET A=INKEY(3)
9Ø VDU 31,X,Y,146,16Ø,16Ø,252,16Ø,16Ø
1ØØ VDU 31,X,Y+1,146,232,236,189,174,
    16Ø
11Ø VDU 31,X,Y+2,146,162,238,177,16Ø,
    16Ø
12Ø VDU 31,X,Y+3,146,24Ø,25Ø,162,18Ø,
    16Ø
13Ø VDU 31,X,Y+4,146,161,16Ø,16Ø,245,
    16Ø
135 LET A=INKEY(3)
14Ø NEXT X
```

Si noti, che non è stata impiegata la tecnica di cancellazione 'sostitutiva', spiegata precedentemente. Il codice di controllo 146 opera automaticamente questa funzione, mentre l'omino 'corre'.

Nel caso del millepiedi, viceversa, essa è necessaria, e così si inseriscono opportunamente gli spazi nel programma:

```
1Ø CLS
2Ø VDU 23;82Ø2;Ø;Ø;Ø;
3Ø FOR X=1 TO 35
4Ø PRINT TAB(X,1Ø);“□OOO<”
5Ø PRINT TAB(X,9);“□)))”
6Ø PRINT TAB(X,11);“□)))”
7Ø A=INKEY(1Ø)
8Ø PRINT TAB(X,1Ø);“□OOO<”
9Ø PRINT TAB(X,9);“□(((”
1ØØ PRINT TAB(X,11);“□(((”
11Ø A=INKEY(1Ø)
12Ø NEXT X
```

La creazione del millepiedi, sul Commodore, si ottiene con i medesimi caratteri alfanumerici visti negli altri casi, ma il metodo per visualizzare differisce.

Si trascriva il seguente programma:

```
)))      )))      )))      )))
000<     000<     000<     000<
)))      )))      )))      )))
```

```
10 PRINT "[CLEAR HOME]"
20 PRINT ")))"
30 PRINT "000<"
40 PRINT ")))"
50 PRINT "[HOME]"
60 PRINT "((("
70 PRINT "000<"
80 PRINT "((("
90 GOTO 10
```

Quando viene eseguito, con un RUN, il programma visualizza un'immagine in rapidissimo movimento, creata per mezzo di speciali caratteri, che si sovrappongono fra loro.

L'istruzione GOTO, infine, fa ritornare l'esecuzione all'inizio, percorrendo un ciclo infinito.

Senza le linee 10 e 50, che usano particolari caratteri grafici e di controllo del Commodore, inseriti nelle PRINT, il programma non riuscirebbe a funzionare regolarmente.

Il simbolo HOME (una S su sfondo invertito) riporta il cursore nella posizione in alto a sinistra dello schermo. Ogni successiva visualizzazione inizia da questa posizione, cosicché i caratteri delle linee 60 e oltre si sovrappongono, cancellandoli, a quelli preesistenti.

Il simbolo CLEAR HOME (un cuore su sfondo invertito), alla linea 10, fa qualcosa in più: dopo aver riportato 'a casa' (HOME) il cursore, ripulisce lo schermo.

RALLENTIAMO L'AZIONE

Il movimento del millepiedi può essere facilmente rallentato, per essere più realistico, con un ciclo FOR ... NEXT. Si aggiunga la linea:

```
45 FOR T=1 TO 100: NEXT
```

Lanciando il programma con un RUN, il movimento risulta adesso migliorato. Il ciclo FOR ... NEXT agisce da semplice contatore (da 1 a 100), prima che l'esecuzione passi alla linea 50.

Intervenendo sul valore massimo del ciclo, si può variare la velocità del movimento.

Si provi anche a spostare la posizione del ciclo FOR ... NEXT, assegnandogli un numero di linea 15. Si otterrà che lo schermo viene ripulito e, solo dopo una pausa, vengono velocemente alternate le due immagini.

Ciò dimostra, in parte, perché sia meglio usare HOME al posto di CLEAR/HOME, in programmi di questo tipo.

Il movimento può essere ancora migliorato se si aggiunge adesso un altro ciclo di ritardo, che operi sulla seconda immagine. Si aggiunga:

```
85 FOR I=1 TO 50: NEXT
```

La pausa introdotta da questo ciclo è sensibilmente minore, per dare più vivacità al movimento delle zampe, ma può esser variata a piacimento.

LO SPOSTAMENTO DELLA FIGURA

Il passo successivo consiste nel far sì che l'insetti si sposti, attraversando lo schermo. Il BASIC del Commodore non possiede la variante PRINT AT, cosicché dobbiamo usare un altro metodo.

In semplici applicazioni, si può ricorrere all'uso della funzione TAB. Questa è sempre seguita da un valore, racchiuso tra parentesi, ad esempio: TAB (15), che posizione il cursore, nel nostro caso, alla colonna 15. In alternativa, la TAB può essere seguita da una variabile numerica (sempre tra parentesi).

Nel nostro programma, adoperiamo quest'ultima soluzione, all'interno di un nuovo ciclo FOR ... NEXT:

```
10 FOR P=0 TO 35
20 PRINT "[CLEAR HOME]" TAB(P) ")))"
30 PRINT TAB(P) "000<"
40 PRINT TAB(P) ")))"
45 FOR I=1 TO 100:NEXT
60 PRINT "[HOME]" TAB(P) "((("
70 PRINT TAB(P) "000<"
80 PRINT TAB(P) "((("
85 FOR I=1 TO 50:NEXT
90 NEXT P
```

Abbiamo eliminato il salto GOTO alla linea 90, adesso rimpiazziato con il nuovo ciclo FOR ... NEXT, nel quale viene incrementato il valore di P, la stessa variabile usata nella TAB. Con ciò si ottiene un graduale spostamento dell'insetto attraverso lo schermo.

Arrivato al lato opposto dello schermo, il millepiedi si ferma. Per farlo ricominciare da capo si può inserire:

```
100 GOTO 10
```

LA GRAFICA SU ROM

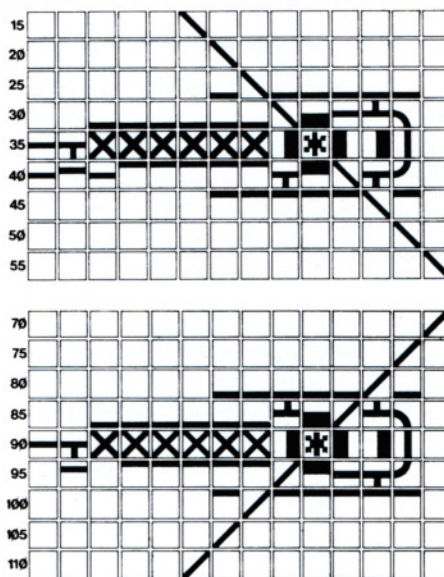
Il Commodore è dotato di una vasta gamma di caratteri grafici predefiniti (nelle memorie ROM), utili per creare nuove ed elaborate forme.

Per potere accedere all'intera gamma di caratteri grafici, è necessario acquistare una certa dimestichezza con il modo 'maiuscolo e grafico' il quale viene ottenuto premendo, contemporaneamente, i tasti C= e SHIFT.

Si provi, a questo punto, a digitare qualche carattere scelto a caso, tenendo come sempre premuto il tasto C= e operando sugli altri tasti.

La visualizzazione di altri caratteri, su sfondo invetito (RVS), si abilita usando contemporaneamente i tasti CTRL e 9 e si disabilita con CTRL e Ø.

La grafica per l'elicottero (figura 3), è ottenuta ricorrendo a questi caratteri, che offrono un piccolo esempio di quanto si possa realizzare.

Ecco il programma:

3. **Come costruire un elicottero. Ogni quadratino è un simbolo grafico ROM**

Qui sotto è riportato un piccolo millepiedi, che può essere facilmente animato sullo Spectrum.

Per cominciare, il programma che ne definisce la forma:

```
)))      )))      )))      )))
000<     000<     000<     000<
)))      )))      )))      )))
```

```
10 PRINT AT 10, 15; "000"
20 PRINT AT 9,15; ")))"
30 PRINT AT 11, 15; ")))"
50 PRINT AT 10,18; "<"
```

Adesso si lanci il programma con RUN. È un sistema piùttosto elaborato, per ottenere una immagine così semplice, ma serve a illustrare due punti importanti.

In primo luogo, offre un'idea abbastanza precisa delle posizioni relative sullo schermo. L'insetto si trova, più o meno centrato sullo schermo, nella linea 1Ø.

In secondo luogo, viene mostrato l'effetto di chiedere al computer di visualizzare più di un carattere nella medesima posizione: i caratteri vengono semplicemente riprodotti nelle locazioni adiacenti. Ecco perché le 'zampe', nella linea 4Ø, sono in posizione 1Ø,18. Le locazioni 1Ø,15; 1Ø,16; 1Ø,17; sono già occupate dal 'corpo' dell'insetto.

Un modo più conveniente di definire l'immagine è di usare una sola linea di programma:

```
10 PRINT AT 10,15; "000<"; AT 9,15 ;")))";
   AT 11,15; ")))"
```

Con sole due linee in più, si ottiene l'animazione:

```
20 PRINT AT 10,15; "000<"; AT 9,15; "(((";
   AT 11,15; "((("
30 GOTO 10
```

Quando il programma viene eseguito, si nota, però, un'eccessiva velocità nel movimento. Il miglior modo di ovviare, consiste nell'introdurre un ciclo di ritardo FOR … NEXT, che conti ad esempio fino a 10 prima di passare all'immagine successiva:

```
15 FOR L= 1 TO 10
17 NEXT L
25 FOR M=1 TO 10
27 NEXT M
```

La durata della pausa si può variare a piacimento: 1 TO 10, oppure 1 TO 20, ecc.

Fin qui abbiamo semplicemente creato un'immagine animata; il movimento viene trattato più avanti (vedere il paragrafo IL MOVIMENTO).

```
 )))      )))      )))      )))
 OOO<     OOO<     OOO<     OOO<
 )))      )))      )))      )))
```

LA GRAFICA SU ROM

Una grafica di qualità migliore si ottiene adoperando i caratteri grafici predefiniti dello Spectrum. Un esempio è riportato in figura 4.

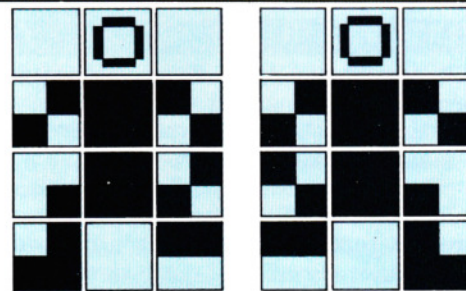
Più avanti riportiamo il programma per intero, ma se non si ha acquistato familiarità con i simboli grafici, conviene procedere gradualmente, creando prima delle immagini statiche:

```
1 PRINT AT 5,15; "O"
2 PRINT AT 6,14; ▞█▚
```

Sullo Spectrum, accedere ai caratteri grafici è semplice, dopo un po' di pratica. Per quelli della linea 2, ad esempio, si premano contemporaneamente i tasti [CAPS SHIFT] e [9]: si entra così nel modo grafico, indicato da una G lampeggiante sullo schermo. Adesso si premano, assieme, [CAPS SHIFT] e [6] per avere un simbolo su sfondo inverso; poi [CAPS SHIFT] e [8]; infine il solo tasto [6]. Prima di digitare i doppi apici, si ricordi di premere il tasto [9], per uscire dal modo grafico.

Ecco il programma completo:

```
10 PRINT AT 5, 14; "□O□"; AT 6, 14;
   "▞█▚"; AT 7, 14; "▗█▛";
   AT 8, 14; "▟□▀
20 PRINT AT 5, 14; "□O□"; AT 6, 14;
   "▞█▚"; AT 7, 14; "▚█▖";
   AT 8, 14; "▀□▙"
30 GOTO 10
```

Di nuovo, inserire un ciclo di ritardo, sia dopo la linea 10, sia dopo la 20, migliora le cose.

4. Un ballerino, in soli 12 quadrati

IL MOVIMENTO

Ecco, infine, un programma che non solo 'anima' il millepiedi, visto precedentemente, ma lo fa anche muovere attraverso lo schermo:

```
10 FOR N=0 TO 27
20 PRINT AT 10,N; "OOO<"; AT 9,N; "(((";
   AT 11,N;"((("
30 PRINT AT 10, N; "□□□"; AT 9,N;
   "□□□"; AT 11, N; "□□□"
40 PRINT AT 10, N; "OOO<"; AT 9, N; ")))";
   AT 11,N;")))"
50 PRINT AT 10, N; "□□□□"; AT 9, N;
   "□□□"; AT 11,N;"□□□"
60 NEXT N
70 GOTO 10
```

Anche in questo programma viene usato un ciclo FOR … NEXT, ma per uno scopo totalmente diverso: prima si è usato per creare delle pause, adesso, invece, per spostare l'immagine sullo schermo, di una posizione (o di un quadrato) alla volta.

Perché alla linea 10 è specificato il valore 27, mentre, in effetti, le colonne dello schermo sono 32?

Per scoprire questo fatto, si sostituisca la linea 10 con:

```
10 FOR N=0 TO 32
```

Un altro indovinello è costituito dalle linee 30 e 50: si provi a cancellarle e si vedrà perché sono necessarie!

I PROSSIMI PASSI

Adesso che abbiamo imparato i rudimenti dell'animazione e del movimento, si può passare a inventare nuove forme e figure di proprio gusto, usando i simboli grafici a disposizione.

Se tuttavia, volessimo utilizzare programmi grafici scritti per altri tipi di computer e che, sfortunatamente non è possibile trascrivere così come stanno, si renderebbe necessario ridisegnare le varie forme, sfruttando i caratteri presenti sul nostro particolare computer.

A

Animazione 26-32

B

BASIC, programmazione
pensa un numero 2-7
i cicli FOR ... NEXT 16-21
Bassa risoluzione,
vedere Grafica
BREAK, *Dragon, Tandy* 4

C

Carro armato,
creazione e controllo
Acorn 11-12
Commodore 64 14-15
Dragon, Tandy 13-14
Spectrum 10
Cassette, scelta 25
CHR$, *Dragon, Tandy* 26-27
Cicli, vedere FOR ... NEXT
CLEAR
Dragon, Tandy 14-27
Spectrum 10
CLOAD, *Dragon, Tandy* 14
CLS, spiegazione 27
CODE, *Spectrum* 8
Codice macchina,
programmazione 8-15
Corridore,
creazione di un, *Acorn* 28-29

D

DATA, istruzione,
Acorn 11
Commodore 64, Vic 20 14
Dragon, Tandy 13
Spectrum 8-9
DIN, presa 22
Dipingere coi numeri 18

E

ELICOTTERO, creazione
Commodore 64 31
ESCAPE, *Acorn* 4

F

FOR... NEXT, cicli 16-21
cicli nidificati 19
creazione di ritardi 17
definizione 16
nei giochi:
Acorn 12
Commodore 64 14
Dragon, Tandy 13
Spectrum 8-9
nei programmi 16-21
nel disegno 18
nella grafica 18-21
nella musica 18
variazioni 19

G

Giochi, indovinelli
Acorn 4-5
Commodore 64, Vic 20 4-5
Dragon, Tandy 3-4
Spectrum, ZX81 3-4
Giochi, programmazione,
animazione 26-32
Giochi, sottoprogrammi, 8-15
GOSUB, *Commodore* 18
GOTO
Acorn 4, 18-21, 28
Commodore 4, 18-21, 30
Dragon, Tandy 4, 18-21, 26
Spectrum, ZX81 4, 18-21, 31
Grafica
bassa risoluzione 26-32
vedere anche:
animazione, movimento,
teletext, ROM grafica, UDG
Griglie, per UDG
vedere UDG

I

IF ... THEN
IF ... THEN ... ELSE
Acorn, Dragon, Tandy 3
INKEY
Acorn 28-29
INPUT, istruzione 3, 4-5
INT 2-3

L

Lancio di missili
Acorn 12
Commodore 64 15
Dragon, Tandy 14
Spectrum 10
LIST, comando 4
LOAD, comando 22-25
esito 23

M

MODE, *Acorn* 28
Movimento
Acorn 28-29
Commodore 64, Vic 20 30-31
Dragon, Tandy 26-27
Spectrum, ZX81 31-32

N

NEW
Acorn 11, 23
Commodore 64, Vic 20 15, 23
Dragon, Tandy 13, 23
Spectrum, ZX81 10, 23
Numeri casuali 2-7
Numeri, dipingere coi 18

P

PMODE, *Dragon, Tandy* 12
POKE
Commodore 64 15
Dragon, Tandy 13
PRINT AT
Dragon, Tandy 26-27
Spectrum, ZX81 8-9, 31-32
PRINT TAB
Acorn 11, 28
Commodore 64, Vic 20 30
Programmi
BASIC 8
interruzione con
BREAK 4, 7, 11
numerazione delle linee 7
punteggiatura 4
rallentare l'esecuzione 17
PSET, *Dragon, Tandy* 13
Punteggiatura,
nella programmazione 4

R

RAM 25
RANA,
creazione e controllo
Acorn 12
Commodore 64 15
Dragon, Tandy 14
Spectrum 10
RANDOMIZE 2
Registratore a cassette,
scelta 24
RND, funzione 2-7
Acorn 2, 3, 4-5
Commodore 64 2, 3, 4, 5, 6, 7
Dragon, Tandy 2, 3, 4, 6, 7
Spectrum, ZX81 2, 3, 4, 6, 7
ROM, grafica
Acorn 28, 29
Commodore 64, Vic 20 31
Dragon, Tandy 26, 27
Spectrum, ZX81 31, 32
RVS, Commodore 31

S

SAVE
comando 22-25
preparazione 22
verifica 24-25
Simboli aritmetici 6
Sprite,
definizione e uso su
Commodore 64 14, 15
STEP 17, 21
STOP, *Spectrum, ZX81* 4
Stringhe, vedere Variabili

T

Tabelle,
di moltiplicazione 5-7
Teletext, grafica *BBC* 28

U

UDG, definizione 8
UDG, griglie per
Acorn 11
Dragon, Tandy 13
Spectrum 8-9

V

Variabili,
nomi per 17
stringa 4-5
uso 3
VDU, comando
Acorn 28-29
VERIFY, comando 24

CORSO PRATICO DI PROGRAMMAZIONE PER LAVORARE E DIVERTIRSI COL COMPUTER

INPUT è un corso di programmazione completamente nuovo e unico nel suo genere che ti introduce subito nel mondo dei computer insegnandoti a programmare con praticità e fantasia.
Tutte le settimane troverai programmi nuovi che ti aiuteranno ad acquisire una padronanza completa del tuo home computer e a migliorare gradualmente il tuo livello operativo.

Diventare esperti è sempre entusiasmante e divertente, con INPUT. Nelle sue 5 Sezioni troverai tutto quello che ti occorre per conoscere sempre meglio le potenzialità del tuo computer.

Nella sezione **PROGRAMMAZIONE BASIC** imparerai il linguaggio di programmazione oggi più diffuso.

Nella rubrica **APPLICAZIONI** scoprirai i molteplici usi dei computer nella vita di ogni giorno, pratica e professionale.

Nella sezione **CODICE MACCHINA** apprenderai a comunicare "direttamente" con il tuo computer, risparmiando così tempo e fatica.

Nella rubrica fissa **PERIFERICHE** troverai informazioni su importanti accessori del computer: unità a dischi, stampanti, joystick, ecc.

In **GIOCHI AL COMPUTER** potrai sperimentare sofisticati effetti grafici, sonori e d'animazione, con varianti inedite, creando una quantità di giochi nuovi: un campo tutto aperto alla tua fantasia.

E in più, per ogni argomento e programma, troverai sempre mille suggerimenti, consigli utili, "trucchi" e stratagemmi per imparare i segreti dei programmatori professionisti.

INPUT è stato studiato appositamente per i seguenti computer: Commodore 64, Sinclair Spectrum 16K e 48K, BBC Model B, Acorn Electron, Dragon 32. Molti programmi sono anche adatti per ZX81, VIC 20, Tandy Colour 32K.

INPUT comprende: *52 fascicoli di 32 pagine, in edicola ogni settimana a L. 2800 ciascuno, che formeranno 6 volumi elegantemente rilegati in similpelle con impressioni in argento e pastello • 1664 pagine complessive • 275 temi di programmazione e software • 1560 fra fotografie e disegni, tutti a colori • In ogni fascicolo un indice parziale più un indice generale al termine dell'opera.*

Non dimenticare! Il prossimo appuntamento tra 10 giorni in edicola con il secondo fascicolo.

"INPUT" È UNA PUBBLICAZIONE DE AGOSTINI